Universidade Estadual de Campinas

LCSI
Faculdade de Engenharia Elétrica e de Computação
Departamento de Máquinas, Componentes e Sistemas Inteligentes
Laboratório de Controle e Sistemas Inteligentes

Modelagem Computacional de Dados e Controle Inteligente no Espaço de Estado

Autor: Annabell Del Real Tamariz
Mestre em Engenharia Elétrica - UNICAMP

Orientador: Prof. Dr. Celso Pascoli Bottura
LCSI / DMCSI / FEEC / UNICAMP

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: **Automação**.

Banca Examinadora

Celso Pascoli Bottura, Dr. ..... UNICAMP
Antonio Augusto Rodrigues Coelho, Dr. ..... UFSC
Peterson de Resende, Dr. ..... UFMG
Gilmar Barreto, Dr. ..... UNICAMP
Marcio Luiz de Andrade Netto, Dr. ..... UNICAMP
Marconi Kolm Madrid, Dr. ..... UNICAMP
Paulo Augusto Valente Ferreira, Dr. ..... UNICAMP

Campinas, S.P.
Julho 2005

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Del Real Tamariz, Annabell
D387m Modelagem computacional de dados e controle inteligente no espaço de estado / Annabell Del Real Tamariz. --Campinas, SP: [s.n.], 2005.

Orientador: Celso Pascoli Bottura.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas MIMO. 2. Sistemas de tempo discreto. 3. Sistemas lineares - Identificação. 4. Series temporais. 5. Espaço e tempo. 6. Sistemas inteligentes de controle. 7. Redes neurais (Computação). 8. Modelagem de dados. I. Bottura, Celso Pascoli. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RMS

Titulo em Inglês: State Space Computational Data Modelling and Intelligent Control
Palavras-chave em Inglês: MIMO systems, Discrete-time systems, Time varing linear systems, Subspace methods, Time series, State space, Intelligent control, Neural networks, Data modeling e Gain-scheduling
Área de concentração: Automação
Titulação: Doutora em Engenharia Elétrica
Banca examinadora: Antonio Augusto Rodrigues Coelho, Peterson de Resende, Gilmar Barreto, Marcio Luiz de Andrade Netto, Marconi Kolm Madrid e Paulo Augusto Valente Ferreira.
Data da defesa: 15/07/2005

Resumo

Este estudo apresenta contribuições para modelagem computacional de dados multivariáveis no espaço de estado, tanto com sistemas lineares invariantes como com variantes no tempo. Propomos para modelagem determinística-estocástica de dados ruidosos, o Algoritmo *MOESP_AOKI*. Propomos, utilizando Redes Neurais Recorrentes multicamadas, algoritmos para resolver a Equação Algébrica de Riccati Discreta bem como a Inequação Algébrica de Riccati Discreta, via Desigualdades Matriciais Lineares. Propomos um esquema de controle adaptativo com Escalonamento de Ganhos, baseado em Redes Neurais, para sistemas multivariáveis discretos variantes no tempo, identificados pelo algoritmo *MOESP_VAR*, também proposto nesta tese. Em síntese, uma estrutura de controle inteligente para sistemas discretos multivariáveis variantes no tempo, através de uma abordagem que pode ser chamada **ILPV** (Intelligent Linear Parameter Varying), é proposta e implementada. Um controlador LPV Inteligente, para dados computacionalmente modelados pelo algoritmo *MOESP_VAR*, é concretizado, implementado e testado com bons resultados.

Palavras-chave: Sistemas Variantes no Tempo, Métodos de Subespaço, Identificação de Sistemas Multivariáveis, Séries Temporais Multivariáveis, Modelagem Computacional de Dados, Espaço de Estado, Escalonamento de Ganhos, Redes Neurais, Controle Inteligente, Controle Adaptativo.

Abstract

This study presents contributions for state space multivariable computational data modelling with discrete time invariant as well as with time varying linear systems. A proposal for Deterministic-Estocastica Modelling of noisy data, *MOESP_AOKI Algorithm*, is made. We present proposals for solving the Discrete-Time Algebraic Riccati Equation as well as the associate Linear Matrix Inequality using a multilayer Recurrent Neural Network approaches. An Intelligent Linear Parameter Varying (ILPV) control approach for multivariable discrete Linear Time Varying (LTV) systems identified by the MOESP_VAR algorithm, are both proposed. A gain scheduling adaptive control scheme based on neural networks is designed to tune on-line the optimal controllers. In synthesis, an Intelligent Linear Parameter Varying (ILPV) Control approach for multivariable discrete Linear Time Varying Systems (LTV), identified by the algorithm MOESP_VAR, is proposed. This way an Intelligent LPV Control for multivariable data computationally modeled via the MOESP_VAR algorithm is structured, implemented and tested with good results.

Keywords: Time Varying Linear Systems, Subspace methods, System Identification, Multivariable Time Series, State Space, Gain Scheduling, Neural Networks, Intelligent Control, Adaptive Control.

Caminante, son tus huellas el camino, y nada más;
caminante no hay camino, se hace camino al andar.
Al andar se hace camino, y al volver la vista atrás
se ve la senda que nunca se ha de volver a pisar.
Caminante, no hay camino, sino estrellas en la mar.

"Proverbios y Cantares", Antonio Machado
Poeta Español (1875 - 1939)

À meu Esposo, Raúl
Aos meus Filhos, Natalie e Javier
Aos meus Pais, Annabell e Gerardo
À minha Avó, Celeste
À minha Irmã, Hilda
À meu Sobrinho, Richard

Agradecimentos

Ao meu orientador, Prof. Celso Pascoli Bottura, sou grata pela dedicação na orientação, por criar melhores profissionais, pela energia positiva que transmite, por acreditar na realização deste trabalho, pelos conselhos e pela oportunidade de trabalhar juntos.

Ao Prof. Gilmar Barreto pela ajuda na confeção e revisão deste trabalho, por estar sempre atento e disposto a resolver problemas, pelas sugestões e inúmeros auxílios.

Aos meus dois filhos, que me deram força para terminar este trabalho;

Aos colegas do LCSI: Ginalber Serra, Sergio, Mauricio, Angel Fernando e João Viana pelas críticas, sugestões e inúmeros auxílios.

Aos colegas do LCSI: Eliezer, Glaucio, Erick, Rogério, Lorena, Amilcar, Felipe, André, pela convivência agradável.

Às colegas da FEEC: Alaíde da Silva Ramos e Edna Servidone.

A meu esposo Raúl, que teve muita paciência e compreensão.

À minha família pelo apoio durante esta jornada.

À minha família brasileira Renata, Roberta, Rosemeire, Daniel e Carminha por fazer do Brasil meu pais.

Aos amigos cubanos Sahudy, Yony, Marta, Luis, Eduardo, Odalys, Roberto, Rosendo, Harold, Daynet, Juan, Zoe, Ileana, Alio que sempre ficaram por perto.

À FAPESP, pelo apoio financeiro.

À UNICAMP.

Sumário

Lista de Figuras **xi**

Glossário **xiii**

Lista de Símbolos **xiii**

Trabalhos Publicados Pelo Autor **xiii**

1 Introdução **1**

2 Fundamentos na Modelagem de Dados no Espaço de Estado **7**
2.1 Identificação no Espaço de Estado 8
2.2 Modelagem Determinística no Espaço de Estado 9
2.3 Modelagem Estocástica no Espaço de Estado 11
2.3.1 Realização Estocástica e Operador de Hankel 15

3 Identificação no Espaço de Estado de Sistemas Variantes no Tempo **19**
3.1 Introdução 19
3.2 Notação 20
3.3 Fundamentação 20
3.4 Identificação Variante no Tempo 25
3.4.1 Determinação do espaço coluna de $\mathcal{O}_k$ 25
3.4.2 Determinação das matrizes A e C 28
3.4.3 Determinação das matrizes B e D 28
3.5 Proposta de Algoritmo $MOESP\_VAR$ 29
3.5.1 MOESP Recursivo 30
3.6 Experimentos com **MOESP_VAR** 30
3.6.1 Exemplos 31
3.7 Modelagem Determinística-Estocástica no Espaço de Estado 36
3.8 Proposta de Algoritmo para Modelagem Computacional de Dados Ruidosos Multivariáveis 37
3.9 Exemplos de Modelagem de Dados Ruidosos 43
3.9.1 Resultados com N4SID 43
3.9.2 Resultados com MOESP_AOKI 47

4 Soluções Neurais de Equações Algébricas de Riccati **57**
4.1 Equação de Riccati Discreta no Tempo . . . 58
4.2 Equação Dinâmica Neural de Riccati Discreta . . . 59
4.3 Equação Dinâmica Neural de Riccati Contínua . . . 62
4.4 Estrutura da Rede Neural . . . 63
4.5 Implementação e Resultados . . . 63

5 Desigualdades Matriciais Lineares: Proposta de Solução Neural da EARD **71**
5.1 Introdução . . . 71
5.2 Equação de Riccati Discreta no Tempo . . . 73
5.3 Equação Dinâmica Neuro-LMI-Riccati Discreta . . . 74
5.4 Estrutura da Rede Neural . . . 76
5.5 LMI-Neural de Riccati Contínua . . . 76
5.6 Implementações e Resultados das Neuro-LMI-Riccati . . . 77

6 Identificação e Controle Inteligente no Espaço de Estado **83**
6.1 Introdução . . . 83
6.2 Controle Inteligente com EG de Sistema LPV . . . 85
6.2.1 Controlador Seguidor LPV Ótimo . . . 87
6.2.2 Escalonamento de Ganhos Neural . . . 90
6.3 Experimentação e Resultados . . . 94

7 Conclusões **101**

Referências bibliográficas **103**

8 Apêndices **115**
8.1 Derivadas . . . 115
8.2 Formas quadráticas definidas . . . 116
8.2.1 Cálculo diferencial aplicado a formas quadráticas . . . 116
8.3 Definições . . . 117
8.4 Dados dos Experimentos . . . 119
8.4.1 Sistema Variante no Tempo . . . 127

9 Inteligência Computacional **131**
9.1 Objetivos da IA . . . 132
9.1.1 Ramos da IA . . . 132
9.2 Uma Introdução às Redes Neurais Artificiais (RNA) . . . 132
9.2.1 RNA - Inspiração Biológica . . . 135
9.2.2 O Elemento Processador (EP) . . . 136
9.3 Principais Arquiteturas de RNA utilizadas para Modelagem e Controle . . . 137
9.4 Aprendizado em Redes Neurais . . . 138
9.4.1 O processo de aprendizado . . . 138
9.5 Desenvolvimento de Aplicações . . . 139
9.6 Redes Neurais para Identificação e Controle . . . 142

Lista de Figuras

3.1 Sinal de saída para k=1 . . . 32
3.2 Sinal de Saída para dois Intervalos de Identificação . . . 34
3.3 Sinal de Saída para dois Intervalos de Identificação . . . 34
3.4 Sinal de Saída para o quinto Intervalo de Identificação . . . 35
3.5 Sinal de Saída para o sexto Intervalo de Identificação . . . 35
3.6 Sinais das saídas y_k, y_{kc} . . . 45
3.7 Sinais das saídas y_{kr}, y_{krc}, respectivamente . . . 46
3.8 Análise do programa N4SID . . . 46
3.9 Saída do Modelo Determinístico . . . 47
3.10 Saída do Modelo Estocástico . . . 48
3.11 Superposição dos Sinais . . . 49
3.12 Análise do Programa AVW . . . 49
3.13 Saída do Modelo Determinístico . . . 50
3.14 Sinal de saída modelada . . . 51
3.15 Superposição dos Sinais . . . 51
3.16 Análise de Erros do Programa **AVW** . . . 52
3.17 Comparação dos Programas Moesp_Aoki e N4SID . . . 52
3.18 Saída do Modelo Determinístico . . . 53
3.19 Sinal de saída modelado . . . 54
3.20 Superposição dos Sinais obtidos pelo *N4SID* . . . 55
3.21 Nova Análise de Erros do Programa *N4SID* . . . 55

4.1 Resultados da equação neural de Riccati Contínua . . . 64
4.2 Resultados de V(t) e Z(t), para a EARD usando *RNR* . . . 66
4.3 Resultados de K(t), para a EARD usando *RNR* . . . 66
4.4 Resultados de x(t) e u(t), usando *RNR* . . . 67
4.5 Resultados de Z(t) e V(t) usando *RNR* . . . 68
4.6 Resultados de B(t) e K(t) usando *RNR* . . . 68
4.7 Resultados de V(t) com *RNR* . . . 69
4.8 Resultados de x(t) e u(t) usando *RNR* . . . 69

5.1 Trajetórias de V e $\widetilde{R_1}$, respectivamente . . . 79
5.2 Trajetórias de V . . . 80
5.3 Trajetórias de $\widetilde{R_1}$ . . . 80

5.4 Trajetória de $G(P, R_1)$ 81
5.5 Trajetórias de V e $\widetilde{R1}$. 81

6.1 Sistema de Controle Adaptativo Clássico por EG 85
6.2 Arquitetura do Controlador Neural 90
6.3 Rede Neural Feedforward 91
6.4 Esquema de Treinamento do Controlador Inteligente LPV 93
6.5 Trajetória dos elementos de $a2_k$ e das saídas y_k, respectivamente 95
6.6 Trajetórias dos elementos na matriz de saída Y_c 96
6.7 Saída do Sistema Controlado Y_c 97
6.8 Sinal de controle 97
6.9 Treinamento Neural por Backpropagation 98
6.10 Saída do Sistema Controlado Inteligentemente 99
6.11 Sinal de Controle Inteligente 100
6.12 Superficies de ganhos neurais para um conjunto de referências 100

8.1 Sinal de saída para k=4 128
8.2 Sinal de saída para k=7 128
8.3 Sinal de saída para k=10 129

Trabalhos Publicados Pelo Autor

1. A.D.R. Tamariz, C.P. Bottura, "Soluções Neurais de Inequações Matriciais Lineares de Riccati", *VII Simpósio Brasileiro de Automação Inteligente (SBAI 2005)*, São Luís, Maranhão-Brasil, 19-23 Setembro, 2005.

2. A.D.R. Tamariz, C.P. Bottura, "Discrete-Time Algebraic Riccati Inequation Neuro-LMI Solution", *Proceedings of the International Conference on Systems, Man and Cybernetics (IEEE SMC 2005)*, Hawaii-USA, October 10-12 2005.

3. A.D.R. Tamariz, C.P. Bottura, "Soluções Neurais de Equações Algébricas de Riccati", *VII Congresso Brasileiro de Redes Neurais (CBRN 2005)*, Natal, 16 a 19 de outubro de 2005.

4. A.D.R. Tamariz, C.P. Bottura, G. Serra, "Intelligent Gain-Scheduling Control for Multivariable Discrete Linear Time Varying Systems", *Anais do XVIII Congresso Brasileiro de Engenharia Mecânica*, (COBEM'05), Ouro Preto, MG, 6-11 novembro, 2005

5. A.D.R. Tamariz, C.P. Bottura, "Discrete-Time Systems Neuro-Riccati Equation Solution", *Proceedings of the IEEE International Joint Conference on Neural Networks*, (IJCNN05), Montréal, Québec, Canada, July 31-August 4, 2005.

6. A.D.R. Tamariz, C.P. Bottura, G. Barreto, "Iterative MOESP Type Algorithm for Discrete Time Variant System Identification", *Proceedings of the 13th Mediterranean Conference on Control and Automation*, (MED'2005), Limassol, Cyprus, June 27-29, 2005.

7. A.D.R. Tamariz, C.P. Bottura, G. Barreto, "Discrete Time Variant System Identification", *Anais do* 1^{st} *Meeting on Computational Modelling (LNCC)*, Petrópolis, August 9-13, 2004.

8. A.D.R. Tamariz, C.P. Bottura, G. Barreto, "Algoritmo Iterativo do tipo MOESP para Identificação de Sistemas Discretos Variantes no tempo - PARTE I: Formulação", *Anais do Congresso Temático de Aplicações de Dinâmica e Controle da Sociedade Brasileira de Matemática Aplicada e Computacional (SBMAC)*, (DINCON), Série Arquimedes, Volume 2, São Jose dos Campos, SP, Brasil, 18-22 Agosto, 2003.

9. A.D.R. Tamariz, C.P. Bottura, G. Barreto, "Algoritmo Iterativo do tipo MOESP para Identificação de Sistemas Discretos Variantes no tempo - PARTE II: Implementação e Experimentação", *Anais do Congresso Temático de Aplicações de Dinâmica e Controle da Sociedade Brasileira de Matemática Aplicada e Computacional (SBMAC)*, (DINCON), Série Arquimedes, Volume 2, São Jose dos Campos, SP, Brasil, 18-22 Agosto, 2003.

10. C.P. Bottura, A.D.R. Tamariz, G. Barreto, A.F.T. Cáceres, "Parallel and Distributed MOESP Computational System's Modelling", *Proceedings of the* 10^{th} *Mediterranean Conference on Control and Automation*, (MED'2002), Lisboa Portugal, July 9-12, 2002.

11. C.P. Bottura, G. Barreto, M.J. Bordon, A.D.R. Tamariz, "Parallel and Distributed Computational Multivariate Time Series Modeling in the State Space", *Proceedings of the American Control Conference*, (ACC'2002), Anchorage, Alaska, USA, pg 1466-1471, Maio 2002.

12. A.D.R. Tamariz, C.P. Bottura, G. Barreto and J.V. Fonseca Neto, "Parallel and Distributed Multivariable Identification Via the MOESP Approach, *Anais do XVI Congresso Brasileiro de Engenharia Mecânica*, (COBEM'01), Uberlândia, MG, 26-30 novembro, 2001

13. A.D.R. Tamariz, C.P. Bottura, G. Barreto and M.J. Bordon, "An Approach to State Space Computational Modeling and Prediction of Time Series in Parallel and Distributed Computers", *Anais do XVI Congresso Brasileiro de Engenharia Mecânica*, (COBEM'01),Uberlândia, MG, 26-30 de novembro, 2001.

14. A.D.R. Tamariz, C.P. Bottura, G. Barreto and M.J. Bordon, "Parallel and distributed computational data modelling via Verhaegen & Dewilde's subspace method". *Proceedings of the 2000 American Control Conference*, (ACC'2000), June 28-30, Chicago, Illinois, USA, 2000.

15. A.D.R. Tamariz, C.P. Bottura, G. Barreto and M.J. Bordon, "Parallel and distributed State - Space Modeling for Computation of Time Series using Realization theory, *Proceedings of the Third International Symposium on Mathematical Modelling*, Vienna, Austria, February 2-4, 2000.

16. D. Monett, T. Luis, A. Soto, A.D.R. Tamariz, H.-D. Burkhard, K. Bothe, "10 años de cooperación en el tema: Sistemas Inteligentes", Memorias de Diálogos transatlânticos, *Anais da Conferencia Conjunta de la Universidad de La Habana y la Universidad de Humboldt de Berlín*, La Habana, Cuba, Feb-Mar, 2000.

17. C.P. Bottura, A.D.R. Tamariz, G. Barreto, J.V. Fonseca "Sequential and Parallel Algebraic Riccati Equations Solution via ESST on the Schur Method", *Proceedings of the 38th IEEE Conference on Decision and Control*, (CDC'99), Phoenix, Arizona, USA, pp.2739 - 2740, December 1999.

18. C.P. Bottura, G. Barreto, M.J. Bordon, A.D.R. Tamariz, "Tratamento Computacional de Alto Desempenho em Método de Subespaços para Modelagem de Dados", *XV Congresso Brasileiro de Engenharia Mecânica*, (COBEM'99), Águas de Lindóia, São Paulo, Brasil, 22-26 Novembro, 1999.

19. A.D.R. Tamariz, C.P. Bottura, J.V. Fonseca, G. Barreto, "Formas Sequencial e Paralela para Solução da Equação Algébrica de Riccati por um Algoritmo de Schur-Modificado", *Anais do XV Congresso Brasileiro de Engenharia Mecânica*, (COBEM'99), Águas de Lindóia, São Paulo, Brasil, 22-26 Novembro, 1999.

20. A.D.R. Tamariz, C.P. Bottura, J.V. Fonseca e G. Barreto, "A Parallel and Distributed Solution Method for the Riccati Equation Via Stabilized Elementary Similarity Transformation", *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computation*, (SIAM'99), San Antonio, Texas, March 22-29, 1999.

21. A.D.R. Tamariz, C.P. Bottura, G. Barreto, M.J. Bordon, "A High Computational Performance Approach for a Subspace Identification Method", *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, (SIAM'99), San Antonio, Texas, March 22-24, 1999.

Capítulo 1

Introdução

Esta tese de doutorado, como o título sugere, está dividida em duas partes. Na primeira parte consideramos a **Modelagem Computacional de Dados** e na segunda parte o **Controle Inteligente**, objetivando modelar e controlar sistemas dinâmicos lineares **multivariáveis** discretos no tempo no **espaço de estado**.

Por tratar-se de um trabalho envolvendo alguns campos distintos de pesquisas, este texto foi desenvolvido na forma mais concisa possível, apoiando-se quando necessário nas teses [12, 32] também elaboradas no *Laboratório de Controle e Sistemas Inteligentes (LCSI)*, nos apêndices e nas referências bibliográficas citadas.

De forma geral a Modelagem Computacional de Dados no Espaço de Estado para sistemas dinâmicos lineares com múltiplas entradas e múltiplas saídas $(MIMO)$ a partir das medidas de entrada e saída em ambientes ruidosos, é um problema central em modelagem multivariável de séries temporais, processamento de sinais e em identificação, análise e projeto de sistemas de controle. Em termos gerais, este problema é equivalente a encontrar realizações para sistemas e sinais dinâmicos que representem seqüências de dados de entrada-saída em ambiente ruidoso. Por realização, entende-se a determinação de quádrupla de matrizes que representem os dados de entrada-saída com validação aceitável.

Entendemos por *Modelagem de Dados* a Modelagem de Séries Temporais e a Identificação de Sistemas. Tópicos de Identificação têm sido estudados extensamente devido a que a análise e o projeto convencional de sistemas de controle requerem modelos matemáticos suficientemente precisos; a área de Identificação de Sistemas constitui um campo maduro e de grande interesse para os engenheiros de controle.

A construção de modelos é a essência da análise de Séries Temporais e da Identificação de Sistemas, e eles são utilizados para:

1. Descrever suscintamente o comportamento dos dados;

2. Modelar matematicamente sistemas e/ou sinais;

3. Explicar o comportamento de Séries Temporais, bem como o comportamento de Sistemas a partir de suas entradas e saídas;

4. Prever;

5. Controlar;

6. Supervisionar;

7. Otimizar;

8. Diagnosticar.

Muitos dos métodos desenvolvidos para identificação de parâmetros são baseados em modelos paramétricos descritos por equações de estado. Fundamentalmente duas abordagems podem ser classificadas em relação aos elementos básicos na construção da matriz de dados, tal como a matriz de Hankel [58, 12]. A primeira abordagem utiliza a função resposta ao impulso para construir o modelo dos dados no espaço de estado e é classificada como método de identificação no domínio do tempo ou espaço de estado. Já a segunda abordagem utiliza uma matriz função de transferência para realizar o modelo dos dados e então identificar os parâmetros do modelo e é classificada como método de identificação no domínio da freqüência [72].

Em fins da década de 1980 ocorreu o nascimento de um novo tipo de algoritmo para modelagem de séries temporais e para identificação de sistemas lineares no espaço de estado, os chamados "métodos de subespaço", os quais constituem motivação importante para nosso trabalho.

O problema principal tratado na identificação de sistemas no espaço de estado pode ser definido como: *Dado um número de medidas de entradas, u_k, e de saídas y_k, geradas por um sistema desconhecido, determinar a ordem n do sistema e as matrizes A, B, C e D, a menos de uma transformação de similaridade e as matrizes de covariâncias do ruído Q, R e S.*

Modelos matemáticos de sistemas dinâmicos lineares são e serão importantes na maioria das áreas da ciência. Modelos deduzidos a partir de dados experimentais constituem a essência da identificação de sistemas. Uma abordagem clássica através da estimação dos parâmetros do modelo é baseada na idéia da estimação por máxima verosimilhança, que para o caso de sistemas com múltiplas entradas - múltiplas saídas, exige um elevado esforço computacional. Normalmente o objetivo da identificação de sistemas é a obtenção de um modelo com a finalidade de controlar um processo; deste modo o modelo obtido deve representar de forma adequada a saída do processo; isto pode ser obtido minimizando, através de algum critério, a diferença entre a saída verdadeira e a saída estimada do processo, por exemplo, como é o caso dos métodos de erro de predição.

Os métodos de subespaços utilizados neste trabalho são fundamentalmente diferentes e permitem a determinação direta dos estados do sistema através de técnicas de subespaços. As técnicas de subespaços envolvem princípios e conceitos que tornam mais simples a modelagem multivariada. Neste contexto a noção de estado é fundamental e a estimação no espaço de estado tem um papel principal. A teoria de sistemas lineares, sofreu grandes desenvolvimentos no século XX e a abordagem via espaço de estado por Kalman foi de grande impacto [33, 140]. A modelagem de séries temporais, a identificação e a análise de sistemas foram e são as grandes beneficiárias desta teoria, mas em diferentes épocas e abordagens. As teorias de realização determinística e estocástica visando a modelagem de sistemas dinâmicos no espaço de estado e utilizando algoritmos baseados em subespaço, gerados por entradas e saídas passadas, são muito úteis para modelagem de dados, estimação de estado e controle.

Métodos de identificação por subespaço têm provado ser uma excelente alternativa aos métodos de erro de predição clássicos e para a identificação de sistemas lineares multivariáveis de ordem

elevada. Um dos objetivos desta tese é estender as metodologias teóricas individuais de métodos de identificação no espaço de estado e de modelagem de séries temporais no espaço de estado, ou seja, combina-las para resolver problemas de identificação de sistemas multivariáveis ruidosos tanto invariantes como variantes no tempo, de forma acurada.

A partir do modelo obtido determina-se um *Controle Convencional ou um Controle Inteligente* para o sistema sob estudo: *Controle* baseado em modelo. Outra alternativa seria: *Controle Inteligente* não baseado em modelo [95, 173].
O termo *Controle Convencional* é usado para as teorias e métodos desenvolvidos, sobretudo nas décadas passadas, para controlar sistemas dinâmicos, cujo comportamento é fundamentalmente descrito por equações diferenciais ou a diferenças finitas [113], ou seja usado para tratar situações não rígidas em aplicações práticas.

Os conceitos e métodos desenvolvidos em *Teoria de Sistemas de Controle* e as novas técnicas em desenvolvimento no campo da *Inteligência Computacional* [9] como as *Redes Neurais* e os *Algoritmos Genéticos* podem ser convenientemente combinados para o Controle Inteligente de Sistemas Dinâmicos Complexos na presença de incertezas. Estudos de Narendra [105, 103] mostram que a utilização das *Redes Neurais* pode fornecer melhores soluções que técnicas tradicionais de identificação e controle. Seus trabalhos podem ser considerados passos importantes na identificação e controle de sistemas utilizando *Redes Neurais*. Na visão de Harris, [54] pouco ganho se obtém quando aplica-se Controle Inteligente a sistemas lineares invariantes no tempo, na verdade não somente Controle Inteligente. O mesmo não se deve dizer para o caso de sistemas lineares variantes no tempo, tema importante neste estudo.

Intrinsicamente relacionado ao problema de identificação está o problema de *Controle Adaptativo*, cuja motivação principal é muito atraente: um controlador que modifica-se a si mesmo baseado no comportamento da planta controlada, de forma a satisfazer algumas especificações de projeto [64, 8]. O elemento principal deste método de projeto de controladores é o mecanismo de ajuste dos parâmetros do controlador. Entre os tipos principais de técnicas de ajuste destes parâmetros podemos mencionar o Escalonamento de Ganhos (EG), o controle adaptativo baseado num modelo de referência, [111], auto-tunning, self-tunning, pattern recognition, etc.

O Controle Adaptativo é uma importante área para muitos pesquisadores, entre os quais podemos citar [8, 77, 91, 98, 102] e tem sido utilizado principalmente para melhorar o desempenho on-line dos controladores. O desenvolvimento da teoria de controle adaptativo e sua viabilização em microprocessadores conduziu a uma série de aplicações com bons desempenhos em áreas tais como robótica e controle de aeronaves, dentre outras.

O *Controle Inteligente* foi originalmente proposto por Fu, [50], e foi definido como uma abordagem para gerar ações de controle pelo emprego de aspectos de inteligência computacional, pesquisa operacional e sistemas de controle automático. É uma área de aplicação de Inteligência Computacional ao Controle de Sistemas considerada sucessora do controle adaptativo da década de 1970. Estratégias são definidas e buscam ser capazes de alcançar e manter o nível desejado de desempenho na presença de grandes incertezas em sistemas de malha fechada. No controle de sistemas complexos, dentre as dificuldades que aparecem, ressaltaremos as que podem ser classificadas em quatro categorias: complexidade computacional, não-estacionariedade, não-linearidade e incertezas. Aos sistemas de controle capazes de lidar com tais categorias de dificuldades utilizando a inteligência computacional chamaremos *Sistemas de Controle Inteligente*.

O uso da terminologia *Controle Inteligente* agrupa diversas metodologias, [21, 89], combinando

a teoria de controle convencional com técnicas de inteligência computacional baseadas em Redes Neurais (RN), lógica nebulosa, sistemas especialistas, algoritmos genéticos e uma ampla variedade de técnicas de busca e otimização.

Sistemas de controle usando Redes Neurais e/ou Lógica Nebulosa são alternativas viáveis em controle adaptativo. Pesquisas com Redes Neurais para aplicações em controle estão sendo realizadas por alguns pesquisadores, [36, 103, 117, 129]. Alguns trabalhos em projeto de controladores neurais discretos no tempo foram realizados por [85, 86, 15, 128]; até o momento não se tem muitos resultados para o controle em malha fechada de sistemas não-lineares e/ou não-estacionários discretos no tempo usando Redes Neurais multicamadas.

Num sistema de controle realimentado medem-se as saídas do sistema, comparam-se os resultados com as saídas desejadas e então o sinal de erro produzido é utilizado para calcular as entradas de controle do sistema de tal maneira que o erro torne-se pequeno. Sistemas de controle realimentado produzidos pelo homem são responsáveis, por exemplo, pelos avanços que a era aeroespacial têm nos dias de hoje e também são usados no controle industrial, automotivo, etc.

Um dos objetivos proposto nesta tese é aprofundar o estudo do problema de Controle Multivariável e especificamente desenvolver gradualmente métodos numéricos e computacionais para o Controle Inteligente de sistemas multivariáveis variantes no tempo, a partir da teoria de controle ótimo, usando *Redes Neurais Artificiais (RNA)*.

O *Controle Inteligente* é uma área de aplicação da Inteligência Computacional, que procura resolver problemas que ainda não foram cobertos por outros campos de pesquisa. Estratégias podem ser definidas, e buscam ser capazes de alcançar e manter o nível desejado de desempenho em sistemas complexos na presença de incertezas.

Neste estudo:

1. Pesquisamos alternativas para análise, projeto, estruturação e/ou computação para **Modelagem de Dados** e **Controle Inteligente** de sistemas dinâmicos multivariáveis lineares com parâmetros desconhecidos, no espaço de estado, tanto invariantes como variantes no tempo, tanto determinísticos como estocásticos.

2. Desenvolvemos procedimentos computacionais para a **Modelagem de Dados** multivariáveis no espaço de estado com base em subespaço do mesmo, bem como para o seu **Controle Inteligente** baseado em Redes Neurais.

Esta tese de doutorado está estruturada da seguinte maneira.

No capítulo 2 uma introdução com alguns fundamentos essenciais para modelagem computacional de dados no espaço de estado, que inclui tanto aspectos de realizações determinísticas como estocásticas, é apresentada; tais pontos são essenciais para este trabalho, que enfatiza para dados experimentais, propostas e implementações de algoritmos para:

- Modelagem Determinística;
- Modelagem Estocástica;
- Modelagem Determinística-Estocástica.

realizadas nos Capítulos 3 e 6.

O capítulo 3 tem como objetivos:

- Propor e desenvolver um algoritmo iterativo, com as vantagens dos métodos de subespaço, para identificar sistemas variantes no tempo; para isto propomos, formulamos e implementamos um procedimento computacional que chamamos $MOESP\_VAR$, [146], para a identificação no espaço de estado de sistema multivariável linear discreto variante no tempo baseado em método de subespaço do tipo **Multivariable Output-Error State sPace (MOESP)**. Comprovamos a eficiência do algoritmo com experimentação e critério que também propomos.

- Propor e desenvolver, com base nos Capítulos 2 e 3 uma proposta de um método para Modelagem Combinada Determinística-Estocástica de dados no espaço de estado, utilizando a experiência adquirida e algoritmos tratados neste trabalho e em [12, 32], que chamamos Algoritmo *M*OESP_AOKI. Apresentamos também alguns resultados obtidos com uma versão do método de subespaço **Numerical algorithms for Subspace State System IDentification (N4SID)** que faz identificação Determinística-Estocástica no espaço de estado, para comparação com nossa proposta, para a qual também apresentamos os primeiros resultados computacionais.

A segunda parte da tese, trata aspectos do problema de *Controle Multivariável no espaço de estado*, em especial do problema de **Controle Inteligente**. No capítulo 4 propomos uma abordagem que obtém modelos dinâmicos neurais que resolvem a Equação Algébrica de Riccati Discreta (*EARD*) usando uma Rede Neural Recorrente multicamada (*RNR*), comparamos os resultados com os provenientes de outros métodos existentes, em especial com o método de solução proposto durante o desenvolvimento do trabalho [141]. Apoiados nos modelos dinâmicos neurais que descrevem a equação algébrica de Riccati Contínua (*EARC*), fazemos uma implementação computacional que além de calibrar nossa solução neural para este caso, permite validar a proposta de abordagem discreta que fizemos e também implementamos. Apresentamos vários exemplos de aplicação ao problema de projeto de regulador linear quadrático.

No capítulo 5 propostas para resolver via Redes Neurais Artificiais (*RNA*) as inequações matriciais lineares (*LMI*) algébricas de Riccati discreta (*IARD*) e contínua (*IARC*) no tempo são implementadas. Para o caso discreto, nos baseamos em nossa proposta de abordagem que obtém o modelo dinâmico neural que descreve a *IARD* utilizando uma *RNR*, [150]. Para o caso contínuo, apoiados em modelo dinâmico neural que descreve a *IARC*, [87], fazemos uma implementação computacional que além de ajustar nossa solução neural para este caso, permite validar a proposta de abordagem discreta que fizemos e também implementamos. Vários exemplos de aplicação em controle ótimo são apresentados.

No capítulo 6, conceitos e métodos desenvolvidos em *Teoria de Sistemas* e uma das técnicas em desenvolvimento no campo da *Inteligência Computacional*:*Redes Neurais (RN)*, são convenientemente combinadas para realizar Controle Inteligente de Sistemas Dinâmicos Complexos.

Neste Capítulo, propomos um esquema de Controle Adaptativo com Escalonamento de Ganhos baseado em Redes Neurais, para sistemas multivariáveis discretos variantes no tempo. O algoritmo **MOESP_VAR**, proposto no Capítulo 3, é utilizado para a modelagem computacional da planta a partir de dados de entrada-saída, de forma a obter modelos lineares multivariáveis discretos invariantes no tempo em vários pontos de operação. Uma lei de controle linear quadrática seguidor ótima é desenvolvida em malha fechada para cada modelo multivariável identificado, tal que o sistema acompanha uma trajetória desejada num intervalo de tempo dado. Uma abordagem alternativa incluíndo incertezas aditivas aleatórias na dinâmica dos modelos identificados é proposta, resultando num controlador suficientemente robusto para estabilizar a planta variante no tempo. Um escalonador

de ganhos neural é projetado via algoritmo *backpropagation*, para ajustar *on-line* os controladores ótimos projetados. Em síntese, propomos e implementamos uma estrutura de controle inteligente para sistemas discretos multivariáveis variantes no tempo através de uma abordagem que pode ser chamada **ILPV** (*Intelligent Linear Parameter Varying*). Por meio dela concretizamos um controlador LPV Inteligente. Resultados de simulações demonstram a eficiência da metodologia proposta para uma planta multivariável variantes no tempo, importante, por exemplo, em aplicações em controle de aeronaves, de veículos lançadores de satélites e de manipuladores robóticos, dentre outras.

Em síntese são contribuições desta tese de doutorado:

- A proposta e a implementação do algoritmo *MOESP_VAR* para identificação no espaço de estado de sistemas variantes no tempo.
- A proposta teórica de metodologia para resolver problemas de identificação determinística e estocástica combinados num só algoritmo de identificação por subespaço denominado *MOESP_AOKI*, que permite a modelagem computacional de dados ruidosos multivariáveis no espaço de estado.
- A criação de *Benchmarks* para contribuir na solução e no entendimento de problemas de identificação e controle por subespaço e na comparação dos algoritmos para tanto.
- A implementação de algoritmos de identificação por subespaço para realizar Modelagem de dados de sistemas lineares tanto variantes como invariantes no tempo, com a apresentação de resultados e comentários.
- A obtenção de uma única solução definida positiva da Equação Algébrica de Riccati Discreta (*EARD*) usando *RNR*.
- A solução neural da *IARD* por meio de uma *LMI*.
- A proposta e implementação de metodologia original de projeto de um controlador inteligente LPV (Intelligent Linear Parameter Varying) que segue trajetórias com escalonamento neural de ganhos para controle de sistemas lineares discretos multivariáveis variantes no tempo modelados computacionalmente por técnica desenvolvida no Capítulo 3.

Capítulo 2

Fundamentos na Modelagem de Dados no Espaço de Estado

Neste capítulo é considerado o problema de identificação de modelos multivariáveis discretos no tempo no espaço de estado por métodos de subespaço. O objetivo é modelar dados medidos de entrada-saída $\{u_k, y_k\}$ e estado $\{x_k\}$, construindo computacionalmente um modelo discreto invariante no tempo descrito por:

$$\begin{cases} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k \\ y_k &= \mathbf{C}x_k + \mathbf{D}u_k \end{cases} \tag{2.1}$$

A teoria de realização visando a modelagem de sistemas dinâmicos no espaço de estado e utilizando algoritmos baseados em subespaço, gerados por entradas e saídas passadas, tem sido bastante estudada e implementada recentemente e constitui uma alternativa aos tradicionais algoritmos de identificação de sistemas que utilizam métodos baseados no erro de predição [140, 159, 162, 166].

Os métodos tratados e expostos neste trabalho para a obtenção da ordem do sistema e a quádrupla de matrizes podem ser classificados como procedimentos de identificação no espaço de estado baseados em subespaços. Neles o processamento de uma matriz associada com subespaços gerados por colunas de matrizes determinadas por dados de entrada-saída, e construídas numa forma particular, *Matriz de Hankel*, é requerido.

Neste capítulo o tema modelagem computacional de dados no espaço de estado é introduzido. Algumas fundamentações teóricas para modelagem no espaço de estado são expostas de forma concisa. Trataremos nesta tese de três tipos de modelagem de dados que classificaremos como:

- Modelagem Determinística;
- Modelagem Estocástica;
- Modelagem Determinística-Estocástica.

Começamos na seção 2.2 com uma visão geral da Modelagem Determinística, pois este tema está bem desenvolvido em [12]. Damos ênfase à definição do Operador de Hankel e dos parâmetros de Markov. A seguir, na seção 2.3 expomos algumas características importantes para Modelagem Estocástica, pois este tema está apresentado em [32].

2.1 Identificação no Espaço de Estado

Para um sistema linear discreto estacionário multivariável, para qualquer seqüência vetorial de entrada $u(k) \equiv u_k$, a resposta ao estado nulo pode ser descrita por:

$$y_k = \sum_{i=0}^{\infty} G_i u_{k-i} \tag{2.2}$$

sendo y_k o vetor de saída no k-ésimo instante e $G_i \in \Re^{l \times m}$ a matriz resposta à seqüência ao impulso unitário discreto ou meramente a resposta ao impulso discreto do sistema multivariável que analisaremos mais na frente.

Para um sistema discreto multivariável com m entradas, pode-se pensar a entrada $u(k)$ como constituída de uma combinação linear de impulsos $\delta_i = [\ \ldots \ \ 0 \ \ 0 \ \ 1 \ \ 0 \ \ 0 \ \ \ldots\ ]^T$ com 1 apenas na i-ésima componente de entrada, ponderado por:

$$\begin{cases} u_i(0) = 1 & \forall i = 1, 2, \ldots, m \\ u_i(k) = 0 & \forall k = 1, 2, \ldots \end{cases} \tag{2.3}$$

$$u(k) = \sum_{i=1}^{m} u_i(k)\delta_i = u_1(k) \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + u_2(k) \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \ldots + u_m(k) \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{2.4}$$

Podemos fazer uma interpretação em que a resposta ao impulso de um sistema multivariável é obtida fazendo-se a aplicação em paralelo de impulsos unitários δ_i em cada i-ésima entrada, e determinando-se a respectiva resposta ao impulso, $G_i \equiv G(i)$. A matriz resposta ao impulso unitário no instante k pode ser definida como a combinação das respectivas respostas ao impulso para cada uma das entradas do sistema, ou seja:

$$G_k \equiv \begin{bmatrix} G_{k1} & G_{k2} & G_{k3} & \ldots & G_{km} \end{bmatrix} \tag{2.5}$$

onde G_{ki} corresponde à resposta ao impulso no k-ésimo instante de tempo da entrada i.

Se substituímos em (2.2), obtemos uma matriz resposta ao impulso Y com dimensão $l \times m$:

$$Y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} D & 0 & 0 & 0 & 0 \\ CB & D & 0 & 0 & 0 \\ CAB & CB & D & 0 & 0 \\ & \vdots & & & \vdots \\ CA^{k-1}B & \ldots & & CB & D \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_k \end{bmatrix} \tag{2.6}$$

As colunas da matriz da equação (2.6) são os parâmetros de Markov, os quais podem ser obtidos a partir dos dados experimentais com ajuda da função transferência ou resposta ao impulso. Os parâmetros de Markov também podem ser usados para construção de modelos matemáticos para sistemas dinâmicos.

2.2 Modelagem Determinística no Espaço de Estado

Da teoria de realização é conhecido que para o caso determinístico, as matrizes do sistema multivariável (A, B, C, D) podem ser obtidas diretamente da matriz resposta ao impulso:

$$G_k = \begin{cases} 0, & k < 0 \\ D, & k = 0 \\ CA^{k-1}B & k \geq 1 \end{cases} \tag{2.7}$$

Barreto, em sua tese de doutorado, [12], também desenvolvida no Laboratório de Controle e Sistemas Inteligentes (*LCSI*), investiga os fundamentos teóricos para análise, desenvolvimento e implementação de algoritmos para Modelagem Computacional de Dados através de métodos de subespaço, dos quais nos beneficiaremos neste trabalho, e que em parte apresentamos a seguir, e que por outra parte sugerimos consultar [12]. O operador de Hankel H, definido como:

$$H = \begin{bmatrix} G_1 & G_2 & G_3 & \dots \\ G_2 & G_3 & G_4 & \\ G_3 & G_4 & \ddots & \\ \vdots & & & \end{bmatrix} \tag{2.8}$$

é muito importante e útil em vários aspectos de análise e de projeto de sistemas. Sua relação com problemas baseados em dados amostrados é bem conhecida e estabelecida; recentemente tem sido estudado e usado para novas aplicações. Nestas estão incluídos: projetos de filtros digitais, realizações markovianas, modelagem de séries temporais multivariaveis, redução de modelos, realizações balanceadas, entre outras. Em cada um destes casos, a obtenção de função de transferência, impedância ou realização segundo algum critério de otimalidade é reduzida à determinação de alguns valores e vetores singulares de uma matriz de Hankel semi-definida positiva.

A seguir mostraremos, por simplicidade para sistemas monovariáveis, que esta estrutura de Hankel pode ser obtida a partir do conjunto de equações que definem o modelo determinístico no espaço de estado, dado por:

$$\begin{cases} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k \\ y_k &= \mathbf{C}x_k + \mathbf{D}u_k \end{cases} \tag{2.9}$$

Se desenvolvemos esta equação a partir de um instante de tempo inicial k, obtemos que a equação de saída do sistema para $k \geq 0$ pode ser escrita como, [12]:

$$y_k = \begin{cases} Cx_0 + Du_0 & k = 0 \\ CA^k x_0 + \sum_{j=0}^{k-1} CA^{k-j-1}Bu_j + Du_k & k \geq 1 \end{cases} \tag{2.10}$$

A partir do desenvolvimento de (2.9), as saídas são dadas por:

$$\begin{cases} y_{k+1} = & Cx_{k+1} + Du_{k+1} \\ y_{k+2} = & Cx_{k+2} + Du_{k+2} \\ y_{k+3} = & Cx_{k+3} + Du_{k+3} \\ \quad \vdots & \end{cases} \tag{2.11}$$

e sabendo que para a seqüência de entrada u_k, apenas a componente $u_0 = 1$ e que as demais são nulas, a série de equações apresentadas em (2.11), substituindo os estados, podem ser reescritas como:

$$\begin{cases} y_{k+1} = & Cx_{k+1} \\ y_{k+2} = & CAx_{k+1} \\ y_{k+3} = & CA^2x_{k+1} \\ \quad \vdots & \end{cases} \tag{2.12}$$

então, podemos reescrever:

$$\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \end{bmatrix} = \mathcal{O}x_{k+1} \tag{2.13}$$

sendo $\mathcal{O}$ a matriz de observabilidade estendida dada por:

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \end{bmatrix} \tag{2.14}$$

Fazendo o mesmo raciocínio com o vetor de estado x_{k+1}, considerando o vetor de estado inicial $x_0 = 0$, podemos reescrever este como:

$$x_{k+1} = \mathcal{C} \begin{bmatrix} u_k \\ u_{k-1} \\ u_{k-2} \\ \vdots \end{bmatrix} \tag{2.15}$$

onde

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \dots \end{bmatrix} \tag{2.16}$$

é a matriz de atingibilidade estendida ou controlabilidade. Maiores detalhes podem ser consultados em [12].

Substituindo a equação (2.15) na equação (2.13) podemos ver que as observações futuras são expressas em função das entradas atuais e passadas através de uma matriz de Hankel H definida pelo produto das matrizes $\mathcal{O}$ e $\mathcal{C}$:

$$y^+_{k+1} = Hu^-_k \tag{2.17}$$

onde

$$y^+_{k+1} = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \end{bmatrix}, \qquad u^-_k = \begin{bmatrix} u_k \\ u_{k-1} \\ u_{k-2} \\ \vdots \end{bmatrix} \tag{2.18}$$

Os elementos da matriz H são os parâmetros de Markov. O operador de Hankel tem importantes propriedades para modelagem computacional de dados no espaço de estado. Sugerimos consultar [12] para maiores detalhes.

2.3 Modelagem Estocástica no Espaço de Estado

Nesta seção apresentamos apenas algumas características de processos estocásticos estacionários importantes para esta tese e por outra parte fazemos algumas deduções úteis para modelagem estocástica de dados de sistemas lineares discretos invariantes no tempo.

Outras propriedades importantes sobre este tema, são tratadas na tese de doutorado [32], onde os fundamentos teóricos sobre a modelagem estocástica são mais exaustiva e profundamente estudados.

A modelagem de séries temporais multivariadas no espaço de estado requer múltiplos experimentos com seqüencias de dados de entrada. Tal modelo pode ser representado por um sistema linear discreto estocástico multivariável e invariante no tempo, com ruídos brancos na entrada e na saída:

$$\left\{ \begin{array}{lcl} x_{k+1} & = & \mathbf{A}x_k + v_k \\ y_k & = & \mathbf{C}x_k + w_k \end{array} \right. \tag{2.19}$$

onde $\mathbf{x_k} \in \Re^{\mathbf{n}}$ representa o vetor de estado do processo estacionário no sentido fraco; $\mathbf{w_k} \in \Re^{\mathbf{l}}$ e $\mathbf{v_k} \in \Re^{\mathbf{l}}$ são os vetores de ruído com média zero, serialmente não-correlatos, estacionários no sentido fraco; $\mathbf{y_k} \in \Re^{\mathbf{l}}$ é o vetor de observação (saída); $\mathbf{A} \in \Re^{\mathbf{n}\times\mathbf{n}}$ e $\mathbf{C} \in \Re^{\mathbf{l}\times\mathbf{n}}$ são as matrizes a serem determinadas.

Para a realização markoviana, vide Definição 2 na seção 8.3 do Anexo; definida em (2.19), a matriz de covariância está representada pela seguinte expressão.

$$E\left[\left(\begin{array}{c} v_k \\ w_k \end{array} \right) \left(\begin{array}{cc} v_s^T & w_s^T \end{array} \right) \right] = \left(\begin{array}{cc} Q & S \\ S^T & R \end{array} \right) \delta_{k,s} \tag{2.20}$$

onde **E** representa o operador esperança matemática e δ é o delta de Kronecker, cujo valor está definido por:

$$\delta_{k,s} = \left\{ \begin{array}{ll} 1 & se \quad k = s \\ 0 & se \quad k \neq s \end{array} \right. \tag{2.21}$$

Podemos também fazer o estudo de um processo estocástico, estacionário y_k representado pelo mínimo sistema no espaço de estado de dimensão n dado pelo modelo inovativo:

$$\left\{ \begin{array}{lcl} x_{k+1} & = & \mathbf{A}x_k + Ke_k \\ y_k & = & \mathbf{C}x_k + e_k \end{array} \right. \tag{2.22}$$

onde e_k é uma seqüência ruído branco com matriz de covariância dada por $\Delta = E(e_k e_k^T)$.

A seguir apresentamos alguns teoremas e definições úteis no desenvolvimento do capítulo.

Representação de processos gaussianos markovianos

Seja $\{x_k, k = \ldots, -1, 0, 1, \ldots\}$, $x_k \in \Re^n$ um processo estocástico gaussiano markoviano de média nula e covariância definida positiva:

$$E(x_k x_k^T) = \Pi_{k,k} > 0 \tag{2.23}$$

A distribuição de probabilidades de x_k está perfeitamente definida pela sua matriz de covariância:

$$E(x_k x_s^T) = \Pi_{k,s} \tag{2.24}$$

Pelo teorema das projeções, [32]:

$$\begin{aligned} E(x_k/x_s) &= E(x_k x_s^T)E(x_s x_s^T)^{-1}x_s \\ &= \Pi_{k,s}\Pi_{s,s}^{-1}x_s \end{aligned} \tag{2.25}$$

definimos a matriz de projeção, $\Phi(k,s)$, associada ao processo como:

$$\Phi(k,s) = \Pi_{k,s}\Pi_{s,s}^{-1} \tag{2.26}$$

então podemos escrever:

$$E(x_k/x_s) = \Phi(k,s)x_s \tag{2.27}$$

No caso estacionário, temos:

$$E(x_k x_s^T) = \Pi_{k,s} = \Pi_{k-s} = \Pi_\tau \tag{2.28}$$

onde $\tau = k - s$. Então:

$$A_k = \Pi_{k+1,k}\Pi_{k,k}^{-1} = \Pi_1\Pi_0^{-1} = A \tag{2.29}$$

e podemos escrever:

$$x_{k+1} = Ax_k + v_k \tag{2.30}$$

Se chamamos $Q = E(v_k v_k^T)$ e $P = E(x_k x_k^T) = \Pi_0$ e calculamos a matriz de covariância do estado, $E[x_{k+1}x_{k+1}^T]$, obtemos:

$$\begin{aligned} E[x_{k+1}x_{k+1}^T] &= \Pi_0 \\ &= E[(Ax_k + v_k)(Ax_k + v_k)^T] \\ &= AE[x_k x_k^T]A^T + AE[x_k v_k^T] + E[v_k x_k^T]A^T + E[v_k v_k^T] \\ \Pi_0 &= A\Pi_0 A^T + Q \end{aligned} \tag{2.31}$$

substituindo os diferentes termos, obtemos:

$$Q = P - APA^T \tag{2.32}$$

que é uma equação de Lyapunov.

Fazendo a mesma análise para o modelo inovativo representado em (2.22), tem-se:

$$\begin{aligned} E[x_{k+1}x_{k+1}^T] &= \Pi_0 \\ &= E[(Ax_k + Ke_k)(Ax_k + Ke_k)^T] \\ &= AE[x_k x_k^T]A^T + AE[x_k e_k^T]K^T + KE[e_k x_k^T]A^T + KE[e_k e_k^T]K^T \\ \Pi_0 &= A\Pi_0 A^T + K\Delta K^T \end{aligned} \tag{2.33}$$

Continuando o mesmo raciocínio, vamos calcular as covariâncias do estado $\Pi_k = E(x_{t+k}x_t^T)$.

1. Para $k > 0$, da equação (2.30) pode-se escrever as seguintes equações:

$$\begin{aligned} x_{t+k} &= A^k x_t + \sum_{i=0}^{k-1} A^{k-1-i}v_{t+i} \\ \Pi_k &= A^k\Pi_0 = A^k P \end{aligned}$$

2. Para $k < 0$, definimos $l = -k$ e $l > 0$, então podemos escrever $\Pi_k = E(x_{t+k}x_t^T) = \Pi_{-l} = E(x_{t-l}x_t^T)$. Podemos escrever x_t como:

$$\begin{aligned} x_t &= A^l x_{t-l} + \sum_{i=0}^{l-1} A^i v_{t-1-i} \\ \Pi_{-l} &= \Pi_0 A^{lT} = A^k P \end{aligned}$$

como $l = -k$ temos $\Pi_k = P(A^{-k})^T = P(A^T)^{-k}$ para $k < 0$.

De forma geral podemos escrever

$$\Pi_k = \quad E(x_{t+k}x_t^T) = \quad \begin{cases} A^k P, & k \geq 0 \\ P(A^T)^{-k}, & k < 0 \end{cases} \tag{2.34}$$

onde P é uma matriz definida positiva que satisfaz a equação de Lyapunov (2.32).

Vamos calcular a equação de Lyapunov para a matriz de covariância do estado, Π_t. Seja $t = 0$,

$$\begin{aligned} \Pi_0 &= E[x_{k+1}x_{k+1}^T] \\ &= E[(Ax_k + v_k)(Ax_k + v_k)^T] \\ &= AE[x_k x_k^T]A^T + E[v_k v_k^T] \\ \Pi_0 &= A\Pi_0 A^T + Q \end{aligned} \tag{2.35}$$

Para $t = 1$, tem-se:

$$\begin{aligned} \Pi_1 &= E[x_{k+1}x_k^T] \\ &= E[(Ax_k + v_k)x_k^T] \\ &= AE[x_k x_k^T] + E[v_k x_k^T] \\ \Pi_1 &= A\Pi_0 \end{aligned} \tag{2.36}$$

Para $t = 2$, tem-se:

$$\begin{aligned} \Pi_2 &= E[x_{k+2}x_k^T] \\ &= E[(A^2 x_k + Av_k + v_{k+1})x_k^T] \\ &= A^2 E[x_k x_k^T] \\ \Pi_2 &= A^2\Pi_0 \end{aligned} \tag{2.37}$$

Logo podemos generalizar a expressão como:

$$\Pi_t = \begin{cases} A\Pi_0 A^T + Q & t = 0 \\ A^t \Pi_0 & t \geq 1 \end{cases} \tag{2.38}$$

Definimos a matriz de covariância do processo estocástico de saída $\{y_k\}$ da seguinte forma;

$$\Lambda_i = E[y_{k+i}y_k^T] \tag{2.39}$$

que analisamos primeiro para o instante $i = 0$.

$$\begin{aligned} \Lambda_0 &= E[y_k y_k^T] \\ &= E[(Cx_k + w_k)(Cx_k + w_k)^T] \\ &= CE[x_k x_k^T]C^T + CE[x_k w_k^T] + E[w_k x_k^T]C^T + E[w_k w_k^T] \\ \Lambda_0 &= C\Pi_0 C^T + R \end{aligned} \tag{2.40}$$

Definindo a covariância cruzada entre $\{x_k\}$ e $\{y_k\}$ temos:

$$\begin{aligned} G &= E[x_{k+1}y_k^T] \\ &= E[(Ax_k + v_k)(Cx_k + w_k)^T] \\ &= AE[x_k x_k^T]C^T + AE[x_k w_k^T] + E[v_k x_k^T]C^T + E[v_k w_k^T] \\ G &= A\Pi_0 C^T + S \end{aligned} \tag{2.41}$$

Desenvolvendo (2.39) para $i = 1$,

$$\begin{aligned} \Lambda_1 &= E[y_{k+1}y_k^T] \\ &= E[(C(Ax_k + v_k) + w_{k+1})(Cx_k + w_k)^T] \\ &= CAE[x_k x_k^T]C^T + CE[v_k w_k^T] \\ &= CA\Pi_0 C^T + CS \\ \Lambda_1 &= CG \end{aligned} \tag{2.42}$$

Agora para $i = 2$, tem-se

$$\begin{aligned} \Lambda_2 &= E[y_{k+2}y_k^T] \\ &= E[C(A^2x_k + Av_k + v_{k+1}) + w_{k+2}(Cx_k + w_k)^T] \\ &= CA^2E[x_k x_k^T]C^T + CAE[v_k w_k^T] \\ &= CA^2\Pi_0 C^T + CAS \\ \Lambda_2 &= CAG \end{aligned} \tag{2.43}$$

Logo podemos generalizar e escrever:

$$\Lambda_i = CA^{i-1}G \tag{2.44}$$

Isto indica que as covariâncias da saída podem ser consideradas como os parâmetros de Markov do sistema linear invariante no tempo estocástico A, G, C, Λ_0.

Portanto a matriz de covariância da saída do sistema (2.19) tem a forma:

$$\Lambda_i = \begin{cases} C\Pi_0 C^T + R & i = 0 \\ G^T(A^T)^{-i-1}C^T & i < 0; \\ CA^{i-1}G & i \geq 1 \end{cases} \tag{2.45}$$

com $G = A\Pi_0 C^T + S$.

Concluindo temos:

$$\begin{cases} R = & \Lambda_0 - CPC^T \\ Q = & P - APA^T \\ S = & M - APC^T \end{cases} \tag{2.46}$$

O problema de realização estocástica consiste em encontrar um ou mais modelos no espaço de estado através de dados estatísticos do processo, neste caso as covariâncias. O problema pode ser resumido em encontrar as matrizes A, C, M, Λ_0 que satisfazem as equações (2.45) e (2.46). Algoritmos de identificação de subespaço estocásticos calculam modelos no espaço de estado a partir de dados de saída.

2.3.1 Realização Estocástica e Operador de Hankel

Nesta seção, fazemos uma apresentação concisa sobre a Realização Mínima Estocástica do modelo inovativo

$$\begin{cases} x_{k+1} &= \mathbf{A}x_k + Ke_k \\ y_k &= \mathbf{C}x_k + e_k \end{cases} \tag{2.47}$$

A matriz de covariância para a saída $\{y_k\}$ do processo inovativo pode ser expressa em termos da matriz resposta ao impulso e das matrizes do modelo inovativo:

$$\Lambda_i = E[y_{k+i}y_k^T] \tag{2.48}$$

Analisemos primeiro para o instante $i = 0$.

$$\begin{aligned} \Lambda_0 &= E[y_k y_k^T] \\ &= E[(Cx_k + e_k)(Cx_k + e_k)^T] \\ &= CE[x_k x_k^T]C^T + CE[x_k e_k^T] + E[e_k x_k^T]C^T + E[e_k e_k^T] \\ \Lambda_0 &= C\Pi_0 C^T + \Delta \end{aligned} \tag{2.49}$$

Definindo agora

$$\begin{aligned} M &= E[x_{k+1}y_k^T] \\ &= E[(Ax_k + Ke_k)(Cx_k + e_k)^T] \\ &= AE[x_k x_k^T]C^T + AE[x_k e_k^T] + KE[e_k x_k^T]C^T + KE[e_k e_k^T] \\ M &= A\Pi_0 C^T + K\Delta \end{aligned} \tag{2.50}$$

Desenvolvendo para $i = 1$,

$$\begin{aligned} \Lambda_1 &= E[y_{k+1}y_k^T] \\ &= E[(C(Ax_k + Ke_k) + e_{k+1})(Cx_k + e_k)^T] \\ &= CAE[x_k x_k^T]C^T + CKE[e_k e_k^T] \\ &= CA\Pi_0 C^T + CK\Delta \\ \Lambda_1 &= CM \end{aligned} \tag{2.51}$$

Agora para $i = 2$, tem-se

$$\begin{aligned} \Lambda_2 &= E[y_{k+2}y_k^T] \\ &= E[(C(A^2x_k + AKe_k + Ke_{k+1}) + e_{k+2})(Cx_k + e_k)^T] \\ &= CA^2E[x_k x_k^T]C^T + CAKE[e_k e_k^T] \\ &= CA^2\Pi_0 C^T + CAK\Delta \\ \Lambda_2 &= CAM \end{aligned} \tag{2.52}$$

Logo podemos generalizar a expressão como:

$$\Lambda_0 = E(y_k y_k^T) = C\Pi_0 C^T + \Delta \tag{2.53}$$

e

$$\Lambda_i = E(y_{k+i}y_k^T) = CA^i\Pi_0 C^T + CA^{i-1}K\Delta \tag{2.54}$$

A matriz de covariância entre as pilhas de vetores com os dados passados e com os dados futuros da Série Temporal tem a estrutura de uma matriz de Hankel:

$$E[y_{k+1}^{+}y_{k}^{-T}] = E\left[\begin{pmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \end{pmatrix}\begin{pmatrix} y_k^T & y_{k-1}^T & y_{k-2}^T & \dots \end{pmatrix}\right] = \begin{bmatrix} \Lambda_1 & \Lambda_2 & \Lambda_3 & \dots \\ \Lambda_2 & \Lambda_3 & \Lambda_4 & \dots \\ \Lambda_3 & \Lambda_4 & \Lambda_5 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{2.55}$$

A seguir fazemos, da mesma maneira, o desenvolvimento para modelos discretos no espaço de estado representados por:

$$\begin{cases} x_{k+1} = & Ax_k + v_k \\ y_k = & Cx_k + w_k \end{cases} \tag{2.56}$$

Para obter expressões gerais que possibilitem relacionar entradas e saídas de um sistema qualquer a partir de um instante inicial, k, para sistemas discretos representados no espaço de estado pela equação (2.56), observa-se que no instante seguinte tem-se:

$$\begin{cases} x_{k+2} = & Ax_{k+1} + v_{k+1} \\ y_{k+1} = & Cx_{k+1} + w_{k+1} \end{cases} \tag{2.57}$$

Substituindo a equação (2.56) em (2.57) tem-se:

$$\begin{cases} x_{k+2} = & A(Ax_k + v_k) + v_{k+1} \\ y_{k+1} = & C(Ax_k + v_k) + w_{k+1} \end{cases} \Rightarrow \begin{cases} x_{k+2} = & A^2x_k + Av_k + v_{k+1} \\ y_{k+1} = & CAx_k + Cv_k + w_{k+1} \end{cases} \tag{2.58}$$

No instante seguinte tem-se:

$$\begin{cases} x_{k+3} = & Ax_{k+2} + v_{k+2} \\ y_{k+2} = & Cx_{k+2} + w_{k+2} \end{cases} \tag{2.59}$$

que referenciando ao instante inicial resulta em:

$$\begin{cases} x_{k+3} = & A^3x_k + A^2v_k + Av_{k+1} + v_{k+2} \\ y_{k+2} = & CA^2x_k + CAv_k + Cv_{k+1} + w_{k+2} \end{cases} \tag{2.60}$$

E assim sucessivamente.

A solução da equação (2.60) para um dado instante de tempo $k \geq 0$ pode ser escrita em forma generalizada com ruído branco na entrada $\begin{bmatrix} v \\ w \end{bmatrix}$, como:

$$y_k = \begin{cases} Cx_0 + w_0 & k = 0 \\ CA^kx_0 + \sum_{j=0}^{k-1} CA^{k-j-1}v_j + w_j & k \geq 1 \\ CA^kx_0 + \sum_{j=0}^{k-1} \begin{bmatrix} CA^{k-j-1} & I \end{bmatrix}\begin{bmatrix} v_j \\ w_j \end{bmatrix} & k \geq 1 \end{cases} \tag{2.61}$$

Define-se a matriz $\begin{bmatrix} L_{k,j} & G_{k,j} \end{bmatrix}$, com seus respectivos parâmetros de Markov, [12], como:

$$\begin{bmatrix} L_{k,j} & G_{k,j} \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 & I \end{bmatrix} & k = j \\ \begin{bmatrix} CA^{k-j-1} & I \end{bmatrix} & k \geq j+1 \end{cases} \tag{2.62}$$

Como conseqüência dos desenvolvimentos deste capítulo, podemos apresentar o seguinte resultado para uma coleção de saídas, *y*, relacionadas com um processo vetorial ruído branco $\begin{bmatrix} v \\ w \end{bmatrix}$, obtidas a partir do instante 0 até o instante k:

$$y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^k \end{bmatrix} x_0 + \begin{bmatrix} 0 & 0 & & & \cdots & \cdots \\ C & 0 & & & & \\ CA & C & 0 & & & \\ CA^2 & CA & C & & & \\ \vdots & \vdots & & & & \\ CA^{k-1} & CA^{k-2} & CA^{k-3} & \ldots & C & 0 \end{bmatrix} v + \begin{bmatrix} I & 0 & & & \cdots \\ 0 & I & & & \\ 0 & 0 & I & & \\ 0 & 0 & 0 & I & \\ \vdots & & \vdots & \ddots & 0 \\ 0 & 0 & & \cdots & I \end{bmatrix} w \tag{2.63}$$

onde $v = \begin{bmatrix} v_0 & v_1 & v_2 & v_3 & \ldots & v_k \end{bmatrix}^T$ e $w = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 & \ldots & w_k \end{bmatrix}^T$.

Note que *y* pode também ser representado como

$$y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_k \end{bmatrix} = \mathcal{O} x_0 +$$

$$\begin{bmatrix} \begin{bmatrix} 0 & I \end{bmatrix} & 0 & & & \cdots & \cdots \\ \begin{bmatrix} C & 0 \end{bmatrix} & \begin{bmatrix} 0 & I \end{bmatrix} & & & & \\ \begin{bmatrix} CA & 0 \end{bmatrix} & \begin{bmatrix} C & 0 \end{bmatrix} & \begin{bmatrix} 0 & I \end{bmatrix} & & & \\ \begin{bmatrix} CA^2 & 0 \end{bmatrix} & \begin{bmatrix} CA & 0 \end{bmatrix} & \begin{bmatrix} C & 0 \end{bmatrix} & & & \\ \vdots & \vdots & & & & \\ \begin{bmatrix} CA^{k-1} & 0 \end{bmatrix} & \begin{bmatrix} CA^{k-2} & 0 \end{bmatrix} & \begin{bmatrix} CA^{k-3} & 0 \end{bmatrix} & \ldots & \begin{bmatrix} C & 0 \end{bmatrix} & \begin{bmatrix} 0 & I \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} v_k \\ w_k \end{bmatrix} \\ \begin{bmatrix} v_{k+1} \\ w_{k+1} \end{bmatrix} \\ \begin{bmatrix} v_{k+2} \\ w_{k+2} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} v_{k+j-1} \\ w_{k+j-1} \end{bmatrix} \end{bmatrix} \tag{2.64}$$

Segundo a Definição 1 em Anexo, nossa matriz H bloco Hankel de uma matriz resposta ao ruído

branco $\begin{bmatrix} L_{k,j} & G_{k,j} \end{bmatrix}$ é:

$$H = \begin{bmatrix} \begin{bmatrix} L_1 & G_1 \\ L_2 & G_2 \end{bmatrix} & \begin{bmatrix} L_2 & G_2 \\ L_3 & G_3 \end{bmatrix} & \begin{bmatrix} L_3 & G_3 \\ L_4 & G_4 \end{bmatrix} & \dots \\ \begin{bmatrix} L_3 & G_3 \end{bmatrix} & \begin{bmatrix} L_4 & G_4 \end{bmatrix} & \ddots & \\ \vdots & & & \end{bmatrix} \tag{2.65}$$

A matriz de covariância da saída com os vetores de dados passados e futuros da série temporal tem a estrutura da matriz de Hankel para um sistema determinístico :

$$E \begin{bmatrix} y_{k+1}^{+} & y_{k}^{-T} \end{bmatrix} \tag{2.66}$$

onde

$$y_{k+1}^{+} = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \end{bmatrix} \qquad y_{k}^{-} = \begin{bmatrix} y_{k} \\ y_{k-1} \\ y_{k-2} \\ \vdots \end{bmatrix}$$

Conhecendo que $\Lambda_t = E(y_{k+t} y_k^T)$ vamos montar a respectiva matriz de Hankel. Desenvolvendo a expressão (2.66), temos:

$$E \begin{bmatrix} y_{k+1}^{+} & y_{k}^{-T} \end{bmatrix} = \begin{bmatrix} \Lambda_1 & \Lambda_2 & \Lambda_3 & \dots \\ \Lambda_2 & \Lambda_3 & \Lambda_4 & \dots \\ \vdots & \vdots & \vdots & \\ & & & \end{bmatrix} \tag{2.67}$$

assim, obtém-se a matriz de covariância da saída com os vetores de dados passados e futuros da série temporal.

Capítulo 3

Identificação no Espaço de Estado de Sistemas Variantes no Tempo

3.1 Introdução

Uma parte importante das atividades de pesquisa nas áreas de identificação de sistemas e teoria de controle concentra-se no desenvolvimento de esquemas para identificar sistemas dinâmicos lineares invariantes no tempo [90, 139, 159], embora na realidade muitos sistemas físicos e econômicos dentre outros demonstrem comportamento variante no tempo e/ou não-linear. Alguns sistemas não-lineares operam em torno de trajetórias particulares dentro da sua faixa de operação, e podem adequadamente ser descritos como sistemas lineares variantes no tempo, e esta é prática comum em engenharia de sistemas. O desenvolvimento de uma teoria de identificação coerente para esta última classe de sistemas bem como de algoritmos para isto tem grande importância [41, 106, 165].

Com o objetivo principal de desenvolver um algoritmo iterativo, com as vantagens dos métodos de subespaço, para resolver aplicações em tempo real a sistemas lineares variantes no tempo (LVT), neste capítulo da tese, propomos, formulamos e implementamos um procedimento computacional que chamamos $MOESP\_VAR$, [146], para a identificação no espaço de estado de sistema multivariável linear discreto variante no tempo baseado em método de subespaço do tipo **Multivariable Output-Error State sPace (MOESP)**.

Comprovamos a eficácia do $MOESP\_VAR$ com experimentação e critério que também propomos, [147].

Inspirados nas referências [12, 41, 165], fazemos inicialmente um estudo sobre alguns fundamentos teóricos para a identificação de sistemas discretos multivariáveis com base na teoria de realização de sistemas lineares não-estacionários no espaço de estado. Paralelamente, propomos uma estrutura que possa ser utilizada para resolver o problema de identificação de sistemas variantes no tempo a partir de algoritmos já propostos para sistemas invariantes no tempo; isto requer uma redefinição da própria estrutura das matrizes que são processadas com um esquema de identificação para um contexto variante no tempo.

Vamos fazer a identificação usando técnicas de subespaço, particularmente usaremos uma variante particular denominada *MOESP*, para resolver o problema de identificação de modelo variante no tempo no espaço de estado.

Finalmente, com base nas definições expostas no Capítulo 2, elaboramos uma proposta de um

método para Modelagem Combinada Determinística-Estocástica de dados no espaço de estado utilizando a experiência adquirida e algoritmos tratados neste trabalho e em [12, 32] que chamamos algoritmo **Moesp_Aoki**. Apresentamos também alguns resultados obtidos com uma versão do método de subespaço **Numerical algorithms for Subspace State System IDentification (N4SID)** que faz identificação Determinística-Estocástica no espaço de estado, para comparação com nossa proposta, para a qual também apresentamos os primeiros resultados computacionais.

3.2 Notação

Utiliza-se a mesma notação $y = \mathcal{T}u$, utilizada para sistemas lineares invariantes no tempo, [12]. Neste capítulo, para a obtenção do operador de Hankel e o cálculo das matrizes respectivas do modelo, fazemos uma redefinição das próprias matrizes e da estrutura do modelo, que será apresentado na seqüência deste capítulo.

Para um processo estocástico discreto v_k, vamos representar a observação do j-ésimo experimento no instante de tempo k como $v_{j,k}$. O vetor v_k vai ser uma família, no tempo, de vetores $v_{j,k}$ para $j \in [j_0, j_0 + n - 1]$ e $k \in [k_0, k_0 + T - 1]$.

Neste estudo, *Variante no Tempo* significa que as matrizes do sistema A_k, B_k, C_k, D_k podem mudar lentamente com o tempo. Qualitativamente falando, a palavra "lentamente" implica que a matriz muda suave e continuamente, e nunca abrupta ou aleatoriamente. Para fixar idéias, consideremos um intervalo I_j em torno do instante de tempo $k = k_j$, e suponhamos, por exemplo, que a matriz do sistema, A_k, se comporte de forma invariante no tempo durante este intervalo, isto é,

$$A_k = A_{k_j} =: A_j \qquad k \in I_j \tag{3.1}$$

3.3 Fundamentação

Sistemas variantes no tempo fornecem um ponto de vista especial para o estudo das propriedades do mapeamento linear de operadores atuando sobre seqüência de vetores de dados. Um operador linear pode freqüentemente ser decomposto em uma composição de transformações lineares locais nas quais dados intermediários chamados estados são gerados para serem usados em estágios subseqüentes. A transformação global faz o papel do operador entrada-saída ou operador de transferência, enquanto que a decomposição pode ser interpretada como a realização de um esquema computacional no qual pequenas transformações locais são executadas.

O problema de realização, neste contexto, consiste em achar a decomposição do operador original em uma seqüência de operações, cada uma das quais utiliza só dados parciais da seqüência de entrada, gera quantidades intermediárias chamadas estado e produz uma parte da saída. Dado que o operador original é suposto linear, o problema se reduz a achar, para uma dada matriz triangular superior $\mathcal{T}$, a realização $\{A_k, B_k, C_k, D_k\}$ que tenha a matriz dada como operador entrada-saída.

Embora os sistemas dinâmicos sejam descritos ou caracterizados de muitas maneiras, um modo

usual para especificar um sistema dinâmico é a seqüência de resposta ao impulso unitário:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathcal{T} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \tag{3.2}$$

O operador $\mathcal{T}$ pode ser visto como um operador de transferência variante no tempo se sua i-ésima linha contém a resposta ao impulso do sistema para um impulso no instante de tempo i. $\mathcal{T}$ é o operador de transferência do sistema, que mapeia um sinal de entrada $u_k \in R^n$, vetor que representa uma seqüência de entradas

$$u_k = \begin{bmatrix} \ldots & u_{-1} & u_0 & u_1 & \ldots \end{bmatrix}^T \tag{3.3}$$

em um sinal de saída $y_k \in R^n$, vetor que representa uma seqüência de saídas

$$y_k = \begin{bmatrix} \ldots & y_{-1} & y_0 & y_1 & \ldots \end{bmatrix}^T \tag{3.4}$$

Nesta seção vamos considerar a obtenção da parte determinística de uma relação dinâmica entre medidas de entrada e saída do sistema a ser identificado e a ser dada pelo seguinte modelo linear discreto variante no tempo no espaço de estado com múltiplas entradas e múltiplas saídas (MIMO):

$$\begin{cases} x_{k+1} &= A_k x_k + B_k u_k \\ y_k &= C_k x_k + D_k u_k \end{cases} \tag{3.5}$$

onde $u_k \in \mathcal{R}^{m_k}$, $y_k \in \mathcal{R}^{l_k}$, $x_k \in \mathcal{R}^{N_k}$ e as matrizes do sistema têm dimensões adequadas.

A seguir vamos desenvolver expressões gerais que possibilitem relacionar as entradas com o vetor de estado de um sistema qualquer a partir de um instante inicial k, para a equação de estado (3.5).

- Instante $k = 0$

$$x_1 = A_0 x_0 + B_0 u_0 \tag{3.6}$$

- Instante $k = 1$

$$\begin{cases} x_2 &= A_1 x_1 + B_1 u_1 \\ &= A_1 A_0 x_0 + A_1 B_0 u_0 + B_1 u_1 \end{cases} \tag{3.7}$$

- Instante $k = 2$

$$\begin{cases} x_3 &= A_2 x_2 + B_2 u_2 \\ &= A_2 A_1 A_0 x_0 + A_2 A_1 B_0 u_0 + A_2 B_1 u_1 + B_2 u_2 \end{cases} \tag{3.8}$$

  e assim sucessivamente.

Generalizando obtemos

$$x_k = A_{(k-1)} x_0 + \sum_{l=0}^{k-1} A^{(k-l-1)} B_l u_l \tag{3.9}$$

onde de forma geral, $A_{(n)}$ e $A^{(n)}$ representam matrizes de transição que satisfazem:

$$\begin{cases} A_{(0)} = & A_0 \\ A^{(0)} = & I \\ A_{(n)} = & A_n A_{(n-1)} = A_n \dots A_2 A_1 A_0 \\ A^{(n)} = & A_n A^{(n-1)} = A_n \dots A_2 A_1 \end{cases} \tag{3.10}$$

Sejam $j \in [j_0, j_0 + n - 1]$ e $k \in [k_0, k_0 + T - 1]$ onde j_0 indica o início (instante inicial) do primeiro intervalo de experimentação (primeira janela de experimentação), k_0 indica o primeiro instante de tempo do experimento, n indica o número total de experimentos simples (número de janelas) e T indica o número de observações em cada janela, ou seja em cada experimento simples, onde $T \geq n$.

Por exemplo, na interpretação dos parâmetros do problema de identificação o conjunto de índices j, k em $u_{j,k}$ indica a entrada amostrada no instante de tempo k do j-ésimo intervalo de experimentação do sistema (3.11).

Portanto, nosso problema de identificação de sistema variante no tempo fica melhor representado quando consideramos a determinação da seguinte descrição no espaço de estado:

$$\begin{cases} x_{j,k+1} &= A_{j,k} x_{j,k} + B_{j,k} u_{j,k} \\ y_{j,k} &= C_{j,k} x_{j,k} + D_{j,k} u_{j,k} \end{cases} \tag{3.11}$$

baseada nas seguintes seqüências de dados de saída agrupados como:

$$Y_{j,k} = \begin{bmatrix} y^+_{j_0,k_0} & y^+_{j_0,k_0+1} & \cdots & y^+_{j_0,k_0+T-1} \\ y^+_{j_0+1,k_0} & y^+_{j_0+1,k_0+1} & \cdots & y^+_{j_0+1,k_0+T-1} \\ \vdots & \vdots & & \\ y^+_{j_0+n-1,k_0} & y^+_{j_0+n-1,k_0+T-1} & \cdots & y^+_{j_0+n-1,k_0+T-1} \end{bmatrix} \tag{3.12}$$

onde

$$\begin{bmatrix} y^+_{j_0,k_0} & y^+_{j_0,k_0+1} & \cdots & y^+_{j_0,k_0+T-1} \end{bmatrix} = \begin{bmatrix} y_{j_0,k_0} & y_{j_0,k_0+1} & \cdots & y_{j_0,k_0+T-1} \\ y_{j_0,k_0+1} & y_{j_0,k_0+2} & \cdots & y_{j_0,k_0+T} \\ \vdots & \vdots & & \\ y_{j_0,k_0+i-1} & y_{j_0,k_0+i} & \cdots & y_{j_0,k_0+T+i-2} \end{bmatrix} \tag{3.13}$$

e assim sucessivamente para cada intervalo de experimentação simples. Bem como nas respectivas seqüências de entradas $U_{j,k}$ para as mesmas séries de experimentos e sobre o mesmo intervalo de tempo.

Contudo, por simplicidade usaremos a notação

$$A_{j,k} \equiv A_k, \quad B_{j,k} \equiv B_k, \quad C_{j,k} \equiv C_k, \quad D_{j,k} \equiv D_k \tag{3.14}$$

sempre que possível.

A matriz, $Y_{j,k}$, apresenta um conjunto de $(n-1)$ intervalos de experimentação a serem resolvidos. Para maior facilidade de compreensão do problema, vamos analisar apenas um intervalo de experimentação, ou seja $j = 1$.

Para desenvolver expressões gerais que possibilitem relacionar entradas e saídas a partir de um instante inicial, k_0, e fixando um determinado intervalo de experimentação j (uma janela), para sistemas discretos variantes no tempo representados no espaço de estado por:

$$\begin{cases} x_{j,k+1} &= A_k x_{j,k} + B_k u_{j,k} \\ y_{j,k} &= C_k x_{j,k} + D_k u_{j,k} \end{cases} \tag{3.15}$$

observa-se que no instante seguinte tem-se:

$$\begin{cases} x_{j,k+2} = & A_{k+1} x_{j,k+1} + B_{k+1} u_{j,k+1} \\ y_{j,k+1} = & C_{k+1} x_{j,k+1} + D_{k+1} u_{j,k+1} \end{cases} \tag{3.16}$$

Substituindo a equação (3.15) em (3.16) tem-se:

$$\begin{cases} x_{j,k+2} = & A_{k+1}(A_k x_{j,k} + B_k u_{j,k}) + B_{k+1} u_{j,k+1} \\ y_{j,k+1} = & C_{k+1}(A_k x_{j,k} + B_k u_{j,k}) + D_{k+1} u_{j,k+1} \end{cases} \Rightarrow \tag{3.17}$$

$$\begin{cases} x_{j,k+2} = & A_{k+1}A_k x_{j,k} + A_{k+1}B_k u_{j,k} + B_{k+1} u_{j,k+1} \\ y_{j,k+1} = & C_{k+1}A_k x_{j,k} + C_{k+1}B_k u_{j,k} + D_{k+1} u_{j,k+1} \end{cases} \tag{3.18}$$

No instante seguinte, referenciando ao instante inicial resulta em:

$$\begin{cases} x_{j,k+3} = & A_{k+2}A_{k+1}A_k x_{j,k} + A_{k+2}A_{k+1}B_k u_{j,k} + A_{k+2}B_{k+1} u_{j,k+1} + B_{k+2} u_{j,k+2} \\ y_{j,k+2} = & C_{k+2}A_{k+1}A_k x_{j,k} + C_{k+2}A_{k+1}B_k u_{j,k} + C_{k+2}B_{k+1} u_{j,k+1} + D_{k+2} u_{j,k+2} \end{cases} \tag{3.19}$$

e assim sucessivamente.

A solução da equação (3.15) para um dado instante de tempo $k_0 \geq 0$ pode ser escrita como:

$$y_{j,l} = \begin{cases} C_l x_{j,l} + D_l u_{j,l} & l = k_0 \\ C_l A_{(l-1)} x_{j,l} + \sum_{i=0}^{l-1} C_l A^{(l-i-1)} B_i u_{j,i} + D_l u_{j,l} & l \geq k_0 \end{cases} \tag{3.20}$$

Então para uma coleção de saídas, $y_{j_0,l}$, relacionadas com uma coleção de entradas $u_{j_0,l}$, obtidas a partir do instante k_0 até o instante $(k_0 + T - 1)$, para um determinado experimento j_0, tem-se:

$$\begin{aligned} y_{j_0,k_0} = & \begin{bmatrix} C_{k_0} \\ C_{k_0+1}A_{(k_0)} \\ C_{k_0+2}A_{(k_0+1)} \\ C_{k_0+3}A_{(k_0+2)} \\ \vdots \\ C_{k_0+T-1}A_{(k_0+T-2)} \\ \vdots \end{bmatrix} x_{j_0,k_0} + \\ & \begin{bmatrix} D_{k_0} & 0 & & \cdots \\ C_{k_0+1}B_{k_0} & D_{k_0+1} & & \\ C_{k_0+2}A_{k_0+1}B_{k_0} & C_{k_0+2}B_{k_0+1} & & \\ C_{k_0+3}A^{(k_0+2)}B_{k_0} & C_{k_0+3}A_{k_0+2}B_{k_0+1} & D_{k_0+3} & \\ \vdots & \vdots & & \\ C_{k_0+T-1}A^{(k_0+T-2)}B_{k_0} & C_{k_0+T-1}A^{(k_0+T-3)}B_{k_0+1} & \cdots & D_{k_0+T-1} \end{bmatrix} u_{j_0,k_0} \end{aligned} \tag{3.21}$$

onde

$$y_{j_0,l} = \left[\begin{array}{ccccccc} y_{j_0,k_0} & y_{j_0,k_0+1} & y_{j_0,k_0+2} & y_{j_0,k_0+3} & \cdots & y_{j_0,k_0+T-1} \end{array} \right] \tag{3.22}$$

e

$$u_{j_0,k_0} = \left[\begin{array}{ccccccc} u_{j_0,k_0} & u_{j_0,k_0+1} & u_{j_0,k_0+2} & u_{j_0,k_0+3} & \cdots & u_{j_0,k_0+T-1} \end{array} \right] \tag{3.23}$$

A equação (3.21), pode ser escrita em forma compactada através do modelo:

$$Y_H = \mathcal{O}_k \mathcal{X}_H + \mathcal{T}_k U_H \tag{3.24}$$

A matriz Y_H com ln linhas e T colunas tem consecutivos vetores de saída $y_{j,k}$(de dimensão $l \times 1$, onde l é o número de saídas), ordenados da seguinte maneira:

$$Y_H = \left[\begin{array}{cccc} y^+_{j_0,k_0} & y^+_{j_0,k_0+1} & \cdots & y^+_{j_0,k_0+T-1} \\ y^+_{j_0+1,k_0} & y^+_{j_0+1,k_0+1} & \cdots & y^+_{j_0+1,k_0+T-1} \\ \vdots & \vdots & & \vdots \\ y^+_{j_0+n-1,k_0} & y^+_{j_0+n-1,k_0+1} & \cdots & y^+_{j_0+n-1,k_0+T-1} \end{array} \right] \tag{3.25}$$

A matriz U_H com a mesma estrutura da matriz Y_H, contém consecutivos vetores de entrada $u_{j,k}$, ordenados da seguinte maneira:

$$U_H = \left[\begin{array}{cccc} u^+_{j_0,k_0} & u^+_{j_0,k_0+1} & \cdots & u^+_{j_0,k_0+T-1} \\ u^+_{j_0+1,k_0} & u^+_{j_0+1,k_0+1} & \cdots & u^+_{j_0+1,k_0+T-1} \\ \vdots & \vdots & & \vdots \\ u^+_{j_0+n-1,k_0} & u^+_{j_0+n-1,k_0+1} & \cdots & u^+_{j_0+n-1,k_0+T-1} \end{array} \right] \tag{3.26}$$

$\mathcal{X}_H$ contém a seqüência de vetores de estado:

$$\mathcal{X}_H = \left[\begin{array}{cccc} x^+_{j_0,k_0} & x^+_{j_0+1,k_0} & \cdots & x^+_{j_0+n-1,k_0} \end{array} \right]^T \tag{3.27}$$

onde

$$x^+_{j_0,k_0} = \left[\begin{array}{cccc} x_{j_0,k_0} & x_{j_0,k_0+1} & \cdots & x_{j_0,k_0+T-1} \end{array} \right] \tag{3.28}$$

ou seja teremos um valor para cada estado no instante inicial k_0 em cada intervalo de experimentação realizado.

$\mathcal{O}_k$ é uma matriz com estrutura semelhante à da matriz de observabilidade, dada por:

$$\mathcal{O}_k = \left[\begin{array}{c} C_{k_0} \\ C_{k_0+1} A_{(k_0)} \\ C_{k_0+2} A_{(k_0+1)} \\ C_{k_0+3} A_{(k_0+2)} \\ \vdots \\ C_{k+T-1} A_{(k_0+T-2)} \end{array} \right] \tag{3.29}$$

e finalmente,$\mathcal{T}_k$ é uma matriz triangular inferior dada por:

$$\mathcal{T}_k = \begin{bmatrix} D_{k_0} & 0 & & \cdots & 0 \\ C_{k_0+1}B_{k_0} & D_{k_0+1} & & & 0 \\ C_{k_0+2}A_{k_0+1}B_{k_0} & C_{k_0+2}B_{k_0+1} & D_{k_0+2} & & 0 \\ C_{k_0+3}A^{(k_0+2)}B_{k_0} & C_{k_0+3}A_{k_0+2}B_{k_0+1} & C_{k_0+3}B_{k_0+2} & D_{k_0+3} & 0 \\ \vdots & & & & 0 \\ C_{k_0+T-1}A^{(k_0+T-2)}B_{k_0} & C_{k_0+T-1}A^{(k_0+T-3)}B_{k_0+1} & C_{k_0+T-1}A^{(k_0+T-4)}B_{k_0+2} & \cdots & D_{k_0+T-1} \end{bmatrix} \tag{3.30}$$

As matrizes das equações (3.29) e (3.30), propiciam uma representação muito importante para estudar-se as propriedades de uma realização no espaço de estado a partir das seqüências multivariadas de entrada-saída, bem como para o desenvolvimento de algoritmos de identificação multivariável recursivos no espaço de estado, como o exposto neste trabalho.

3.4 Identificação Variante no Tempo

3.4.1 Determinação do espaço coluna de $\mathcal{O}_k$

A matriz U_H para o caso invariante no tempo, é idêntica a matriz U_H do caso variante no tempo se fixarmos um experimento j, ou seja, supomos que temos dados de um único experimento. Logo, para resolver o caso discreto variante no tempo só devemos supor, por enquanto, que temos dados de um único experimento.

Projeções Ortogonais

Quando não existe ruído nos dados de entrada-saída, as matrizes Y_H e U_H podem ser representadas como na equação (3.24). Baseados nisto, vamos determinar o espaço coluna da matriz $\mathcal{O}_k$ para cada instante de tempo, k.

Uma idéia básica para modelagem de dados em subespaços é recuperar o termo $\mathcal{O}_k\mathcal{X}_k$ da equação (3.24); para isto podemos fazer a projeção de cada termo desta equação no complemento ortogonal $U_H^{\perp}$, vide [12] para detalhes:

$$Y_H|U_H^{\perp} = \mathcal{O}_k\mathcal{X}_k|U_H^{\perp} + \mathcal{T}_k U_H|U_H^{\perp} \tag{3.31}$$

donde

$$Y_H|U_H^{\perp} = \mathcal{O}_k\mathcal{X}_k|U_H^{\perp} \tag{3.32}$$

que possibilita a obtenção da matriz de seqüência de estados $\mathcal{X}_k$ e da matriz de observabilidade $\mathcal{O}_k$, através de técnicas de decomposição matricial. Por exemplo, se utilizarmos decomposição em valores singulares (SVD) e/ou QR, ter-se-á:

$$Y_H|U_H^{\perp} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U_1\Sigma_1^{1/2}\Sigma_1^{1/2}V_1^T = QR \tag{3.33}$$

Desta forma podemos concluir que

$$\mathcal{O}_k = U_1\Sigma_1^{1/2} \tag{3.34}$$

e então inferir que o espaço coluna de $\mathcal{O}_k$ é igual ao espaço coluna de U_1 e que:

$$\mathcal{X}_k|U_H^{\perp} = \Sigma_1^{1/2} V_1^T \tag{3.35}$$

Para resolver o problema de modelagem de dados precisa-se estimar a ordem do modelo e identificar as matrizes do modelo no espaço de estado. A ordem n do sistema pode ser determinada, pelo número de valores singulares de $Y_H|U_H^{\perp}$, ou seja, pela dimensão de Σ_1.

Tendo calculado uma extensão do espaço coluna de $\mathcal{O}_k$, nós exploramos sua propriedade de invariância ao deslocamento para calcular a matriz A_k do sistema. Suponha que as colunas da matriz U_H são uma extensão do espaço coluna de $\mathcal{O}_k$, então existe uma matriz não-singular quadrada T_1 que satisfaz, [162]:

$$U_H = \mathcal{O}_k T_1 \tag{3.36}$$

Dado que

$$\mathcal{O}_k^{(1)} A_{k_0} = \begin{bmatrix} C_{k_0} \\ C_{k_0+1} A_{k_0} \\ C_{k_0+2} A_{(k_0+1)} \\ \vdots \\ C_{k_0+T-1} A_{(k_0+T-2)} \end{bmatrix} * A_{k_0} = \begin{bmatrix} C_{k_0} A_{k_0} \\ C_{k_0+1} A_{(k_0+1)} \\ C_{k_0+2} A_{(k_0+2)} \\ \vdots \\ C_{k_0+T-1} A_{(k_0+T-1)} \end{bmatrix} =: \mathcal{O}_k^{(2)} \tag{3.37}$$

Logo, a matriz $\mathcal{O}_k$ é invariante ao deslocamento. A seguir, vamos combinar as expressões (3.36) e (3.37):

$$U_H^{(1)} A_{T_1} := (\mathcal{O}_k^{(1)} T_1)(T_1^{-1} A_{k_0} T_1) = \mathcal{O}_k^{(2)} T_1 = U_H^{(2)} \tag{3.38}$$

As expressões para calcular as demais matrizes serão explicadas detalhadamente nas seguintes seções. É importante notar que o conhecimento da dimensão do espaço coluna de $\mathcal{O}_k$ tem um papel determinante na construção do conjunto sobredeterminado de equações.

No caso onde calculamos a extensão do espaço linha de $\mathcal{X}_k$, não podemos utilizar a propriedade de invariância ao deslocamente no sentido estrito da definição para calcular as matrizes do sistema. De qualquer forma, podemos explorar a estrutura especial da extensão para calcular a quadrúpla de matrizes do sistema de uma vez só. Para isto, suponhamos que as linhas da matriz $\mathbf{X_k}$ são a extensão do espaço linha de $\mathcal{X}_k$. Então existe uma matriz não-singular quadrada T_2 que satisfaz:

$$\mathbf{X_k} = T_2 \mathcal{X}_k \tag{3.39}$$

A matriz $\mathcal{X}_k$ não é invariante ao deslocamento, de qualquer forma podemos combinar as trajetórias do estado, de entrada e de saída num intervalo de tempo determinado, para qualquer experimento j, como segue:

$$\begin{bmatrix} x_2 & x_3 & \dots & x_k \\ y_1 & y_2 & \dots & y_{k-1} \end{bmatrix} = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_{k-1} \\ u_1 & u_2 & \dots & u_{k-1} \end{bmatrix} \tag{3.40}$$

onde escrevendo em forma compactada tem-se que:

$$\begin{bmatrix} x_1 & x_2 & \dots & x_{k-1} \\ u_1 & u_2 & \dots & u_{k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{X_{k-1}} \\ U_{k-1} \end{bmatrix} \tag{3.41}$$

Finalmente concluimos que quando a matriz $\begin{bmatrix} \mathbf{X_{k-1}} \\ U_{k-1} \end{bmatrix}$ tiver posto completo de linhas, podemos calcular a quadrúpla de matrizes do sistema $(A_{T_2}, B_{T_2}, C_{T_2}, D)$.

Quando a entrada $u_{j_0,k}$ é arbitrária e os parâmetros de Markov na matriz $\mathcal{T}_k$ são conhecidos, a partir de (3.24), podemos escrever:

$$\mathcal{O}_k \mathcal{X}_k = Y_H - \mathcal{T}_k U_H \tag{3.42}$$

e com ajuda da fatoração QR dada a seguir,

$$\mathcal{O}_k \mathcal{X}_k = \begin{bmatrix} R_{21} - \mathcal{T}_k R_{11} | R_{22} \end{bmatrix} \begin{bmatrix} Q_1 \\ \hline Q_2 \end{bmatrix} \tag{3.43}$$

pode-se obter, do produto $\mathcal{O}_k \mathcal{X}_k$, o espaço coluna de $\mathcal{O}_k$ ou o espaço linha de $\mathcal{X}_k$. Como é conhecido, o espaço coluna de $\mathcal{O}_k$ pode ser representado pelo espaço coluna de U_H. Logo para obter resultados satisfátorios, temos que trabalhar com a relação entre as entradas e os estados, ou seja, montar uma matriz conjunta $\begin{bmatrix} U_H \\ \hline \mathcal{X}_H \end{bmatrix}$ que possa ter uma fatoração QR dada por:

$$\begin{bmatrix} U_H \\ \hline \mathcal{X}_H \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ \hline R_{x1} & R_{x2} \end{bmatrix} \begin{bmatrix} Q_1 \\ \hline Q_x \end{bmatrix} \tag{3.44}$$

com R_{11} e R_{x2} matrizes quadradas e inversíveis. Com esta notação, podemos exprimir $\mathcal{O}_k \mathcal{X}_k$ como:

$$\mathcal{O}_k \mathcal{X}_k = \mathcal{O}_k (R_{x1} Q_1 + R_{x2} Q_x) \tag{3.45}$$

Combinando a expressão acima com (3.43), obtemos:

$$\mathcal{O}_k R_{x1} Q_1 + \mathcal{O}_k R_{x2} Q_x = (R_{21} - \mathcal{T}_k R_{11}) Q_1 + R_{22} Q_2 \tag{3.46}$$

De aí em diante, desde que $Q_2 Q_1^T = 0$, $Q_x Q_1^T = 0$ e $Q_1 Q_1^T = I$, obtém-se

$$\mathcal{O}_k R_{x1} = R_{21} - \mathcal{T}_k R_{11} \tag{3.47}$$

e

$$\mathcal{O}_k R_{x2} Q_x = R_{22} Q_2 \tag{3.48}$$

Dado que o posto de R_{22} é igual a n, podemos calcular a decomposição em valores singulares desta matriz como:

$$R_{22} = U_n S_n V_n^T \tag{3.49}$$

Como conclusão podemos dizer que o espaço coluna de R_{22} é igual ao espaço coluna de $\mathcal{O}_k$, e por (3.49) deduzimos que o espaço coluna de R_{22} é igual a U_n . Esta conclusão é a chave principal do algoritmo ordinário MOESP.

3.4.2 Determinação das matrizes A e C

Com a conclusão obtida na seção anterior, podemos dizer que existe uma matriz de transformação não-singular quadrada T_1 tal que:

$$\mathcal{O}_k T_1 = U_H \tag{3.50}$$

Substituindo as expressões acima, temos:

$$\begin{bmatrix} C_{k_0} \\ C_{k_0+1}A_{k_0} \\ C_{k_0+2}A_{(k_0+1)} \\ \vdots \\ C_{k_0+T-1}A_{(k_0+T-2)} \end{bmatrix} * T = \begin{bmatrix} \hat{C}_{k_0} \\ \hat{C}_{k_0+1}\hat{A}_{k_0} \\ \hat{C}_{k_0+2}\hat{A}_{(k_0+1)} \\ \hat{C}_{k_0+3}\hat{A}_{(k_0+2)} \\ \vdots \\ \hat{C}_{k_0+T-1}\hat{A}_{(k_0+T-2)} \end{bmatrix} =: U_H \tag{3.51}$$

A propriedade de invariância ao deslocamento da matriz U_H causa então

$$\begin{aligned} U_H^{(1)}\hat{A} &= U_H^{(2)} \\ \hat{C} &= U_H(1:l,:) \end{aligned} \tag{3.52}$$

3.4.3 Determinação das matrizes B e D

O desenvolvimento algébrico para o cálculo das matrizes B e D também é realizado a partir da fatoração QR da matriz de dados de entrada-saída representada pela equação(3.50), da qual:

$$U_H = R_{11}Q_1 \tag{3.53}$$

Retomando a equação (3.24), podemos reescrever:

$$Y_H = \mathcal{O}_k\mathcal{X}_H + \mathcal{T}_k U_H = R_{21}Q_1 + R_{22}Q_2 \tag{3.54}$$

Como o espaço coluna de R_{22} é igual ao espaço coluna de $\mathcal{O}_k$, se a SVD de $R_{22} = U_n S_n V_n^T$, então haverá uma matriz de transformação, T_1, tal que:

$$U_n = \mathcal{O}_k T_1 \tag{3.55}$$

Substituíndo, temos:

$$U_n T_1^{-1}\mathcal{X}_H + \mathcal{T}_k R_{11}Q_1 = R_{21}Q_1 + U_n S_n V_n^T Q_2 \tag{3.56}$$

Como $U_n^{\perp}U_n = 0$, obtemos da equação acima que:

$$(U_n^{\perp})^T\mathcal{T}_k R_{11} = (U_n^{\perp})^T R_{21} \tag{3.57}$$

Como R_{11} tem inversa, a partir da matriz $\mathcal{T}_k$ podemos calcular as matrizes B e D do sistema considerado.

3.5 Proposta de Algoritmo $MOESP\_VAR$

Uma característica básica das diferentes abordagems MOESP é a utilização da fatoração QR. Com isto conseguimos revelar matrizes com espaços linha ou coluna especialmente estruturados.

Nesta tese nos aprofundamos no estudo do esquema denominado MOESP ordinário, o qual utiliza a fatoração QR de uma matriz composta $\begin{bmatrix} U_H \\ Y_H \end{bmatrix}$, e uma outra matriz com linhas ortogonais, para recuperar uma matriz com espaço coluna igual ao espaço coluna de $\mathcal{O}_k$.

Em [161] um algoritmo recursivo rápido para o problema MOESP com matrizes variantes no tempo é apresentado. Variações no algoritmo MOESP podem resultar da forma de realizar os cálculos na fatoração QR, ou seja, quando um novo par amostrado de entrada/saída $\{u_{j,k}, y_{j,k}\}$ é dado, uma atualização parcial da fatoração QR poderia, por exemplo, ser executada. Este novo par entrada/saída determina uma coluna adicional não processada, nas matrizes de Hankel U_H e Y_H.

A principal vantagem deste algoritmo é não fazer uso de nenhuma técnica de atualização da SVD, pois esta demandaria um elevado esforço computacional neste processo.

Nossa proposta de algoritmo de identificação trata um sistema variante no tempo como um conjunto de modelos invariantes no tempo. Desta forma a identificação do sistema variante no tempo consistirá de um conjunto de n modelos invariantes no tempo, que descreverão o sistema para o experimento definido.

Passos do Algoritmo

Por brevidade explicamos apenas o cálculo da matriz do sistema A_k.

Para seqüências de entradas $\begin{bmatrix} u_{k_0} & u_{k_0+1} & \dots & u_{k_0+T-1} \end{bmatrix}$ e saídas $\begin{bmatrix} y_{k_0} & y_{k_0+1} & \dots & y_{k_0+T-1} \end{bmatrix}$:

1. Construir as matrizes Hankel Y_H e U_H definidas em (3.25) e (3.26) respectivamente.

2. Executar a compressão dos dados via fatoração QR:

$$\left[\begin{array}{c} U_H \\ \hline Y_H \end{array}\right] = \left[\begin{array}{c|c} R_{11} & 0 \\ \hline R_{21} & R_{22} \end{array}\right] \left[\begin{array}{c} Q_1 \\ \hline Q_2 \end{array}\right] \tag{3.58}$$

3. Calcular a SVD de R_{22} dada por:

$$R_{22} = \left[\begin{array}{c|c} U_H & U_H^{\perp} \end{array}\right] \left[\begin{array}{c|c} \Sigma_n & 0 \\ \hline 0 & \Sigma_2 \end{array}\right] \left[\begin{array}{c} V_n^T \\ \hline (V_n^{\perp})^T \end{array}\right] \tag{3.59}$$

   Foi demonstrado que o espaço coluna da matriz U_n é igual ao espaço coluna da matriz $\mathcal{O}_k$, [161, 12]. Logo, para uma matriz T_1 quadrada não singular, a relação entre os espaços coluna pode ser expressa como:

$$\mathcal{O}_k = U_H T_1 \tag{3.60}$$

4. Resolver o conjunto de equações

$$U_H^{(1)} A_k = U_H^{(2)} \tag{3.61}$$

   onde $U_H^{(1)}$ e $U_H^{(2)}$ denotam as primeiras e últimas $l(i-1)$ linhas de U_H respectivamente.

A equação (3.61) exprime a propriedade de invariância ao deslocamento satisfeita pelo espaço coluna da matriz U_H ou pelo espaço coluna da matriz $\mathcal{O}_k$.

A seguir, nos concentramos numa atualização recursiva do algoritmo principal, específicamente nos passos 2 e 3 descritos acima.

3.5.1 MOESP Recursivo

Como nossa proposta de algoritmo recursivo MOESP supõe pequenas variações, num intervalo prédefinido de operação, nas matrizes do sistema; desenvolvemos um esquema recursivo aplicável a sistemas que varian lentamente no tempo.

Atualização da fatoração QR

Suponhamos que as medidas $\left[\begin{array}{cccc} u_{j_0,k_0} & u_{j_0,k_0+1} & \dots & u_{j_0,k_0+T-1} \end{array}\right]^T$ e $\left[\begin{array}{ccc} y_{j_0,k_0} & y_{j_0,k_0+1} & y_{j_0,k_0+T-1} \end{array}\right]^T$ já foram processadas pelo algoritmo. Seja a fatoração obtida denotada como:

$$\left[\begin{array}{cc} R_{j_0,11} & 0 \\ R_{j_0,21} & R_{j_0,22} \end{array}\right] \left[\begin{array}{c} Q_{j_0,1} \\ Q_{j_0,2} \end{array}\right] \tag{3.62}$$

onde o índice j_0 representa o conjunto de medidas de entrada-saída processado no experimento simples mais recente.

Suponhamos que durante este intervalo de tempo $[j_0, j_0 + n - 1]$ o modelo no espaço de estado é invariante e igual à

$$\left\{\begin{array}{lcl} x_{k+1} & = & A_{j_0}x_k + B_{j_0}u_k \\ y_k & = & C_{j_0}x_k + D_{j_0}u_k \end{array}\right. \tag{3.63}$$

Logo a equação em forma compactada que relaciona as diferentes matrizes de dados, pode ser representada por:

$$Y_H = \mathcal{O}_k\mathbf{X}_H + \mathcal{T}_{j_0,k}U_H \tag{3.64}$$

3.6 Experimentos com MOESP_VAR

Para o algoritmo numérico, definimos um número n de intervalos de experimentação que vão ter matrizes diferentes. Logo, o novo programa **MOESP_VAR** proposto neste trabalho, terá que ser executado n vezes com T amostras por janela, para conseguir determinar os n conjuntos de matrizes que definem o sistema para cada experimento.

Seja L_j um inteiro específico para o intervalo de experimentação I_j dado por:

$$I_j = \left[\begin{array}{cc} k_j - L_j, & k_j + L_j \end{array}\right] \tag{3.65}$$

com $L_j = v * \Delta$ e $\Delta = S * \Delta_t$, onde v e S são inteiros fixados adequadamente e Δ_t é o período de amostragem.

O instante de tempo de identificação k_j (correspondente ao centro de cada intervalo I_j) é determinado pela recorrência como $k_{j+1} = k_j + \Delta$, k_0 dado. Os diferentes valores de K_{j+1} calculados forman o vetor G.

Um *Benchmark* de exemplo adaptado de [106] com variação suficientemente lenta dos seus parâmetros, é executado para mostrar a eficiência do método proposto e implementado neste trabalho.

$$A_k = \begin{bmatrix} a_k & b_k \\ 1 & -1 \end{bmatrix} \tag{3.66}$$

onde

$$\begin{aligned} a_k = & \quad -\tfrac{1}{2}(k/2500)^2 + \tfrac{1}{2}(k/2500) \\ b_k = & \quad -\tfrac{1}{16}(k/2500)^4 - \tfrac{1}{8}(k/2500)^3 - \tfrac{13}{16}(k/2500)^2 - \tfrac{3}{4}(k/2500) - \tfrac{1}{2} \end{aligned}$$

As outras matrizes do sistema são consideradas constantes:

$$B_k = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}; \qquad C_k = \begin{bmatrix} 1 & 3 \\ 1 & 2 \end{bmatrix} \tag{3.67}$$

$$D_k = \begin{bmatrix} 1 & 3 \\ 1 & 1 \end{bmatrix}. \tag{3.68}$$

A entrada do sistema u_k é escolhida de forma aleatória e é mantida a mesma para os experimentos desta tese (**??**), e deve excitar adequadamente o sistema. No procedimento computacional fazemos o cálculo de $y_{j,k}$ para cada valor de k começando em $k = k_j$.

Nesta seção apresentamos alguns dos resultados obtidos com o $MOESP\_VAR$. Este procedimento foi desenvolvido no *LCSI*, como uma proposta de solução para identificar sistemas lineares variantes no tempo utilizando algoritmos já tratados para sistemas lineares invariantes no tempo.

3.6.1 Exemplos

Exemplo 1: Uma seqüência de entradas-saídas, com vetor de entrada multivariada de 2 elementos e vetor de saída multivariada de 2 elementos, é obtida a partir de uma realização *benchmark* no espaço de estado, para o modelo linear variante no tempo apresentado em (3.11), com as matrizes $A_k \in \mathcal{R}^{2\times2}; B_k \in \mathcal{R}^{2\times2}; C_k \in \mathcal{R}^{2\times2}$ e $D_k \in \mathcal{R}^{2\times2}$, representantes do conjunto de dados original e apresentadas acima, lembrando que a matriz A_k varia lentamente em cada instante de tempo; seu valor para $k = 1$ é:

$$A_1 = \begin{bmatrix} 0.0002 & -0.5003 \\ 1.0000 & -1.0000 \end{bmatrix}.$$

Através da modelagem computacional utilizando o programa $MOESP\_VAR$ para um conjunto de 100 amostras de valores entrada-saída, com $K_0 = 1, v = 20, S = 15, \Delta_t = 0.01sec$, obtivemos o seguinte modelo no espaço de estado para $k = 1$:

$$Ac_1 = \begin{bmatrix} -0.7714 & 1.2353 \\ -0.2622 & -0.2284 \end{bmatrix}; \qquad Bc_1 = \begin{bmatrix} -8.8865 & -3.9576 \\ 7.3860 & -3.1365 \end{bmatrix}$$

$$Cc_1 = \begin{bmatrix} -0.5723 & -0.5532 \\ -0.3880 & -0.4669 \end{bmatrix} \qquad Dc_1 = \begin{bmatrix} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{bmatrix}.$$

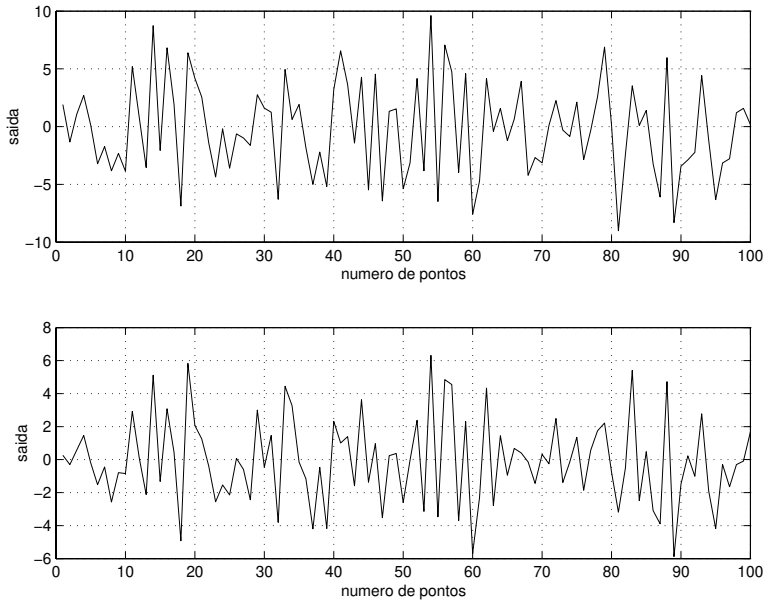

Fig. 3.1: Sinal de saída para k=1

Os resultados podem ser observados na Figura 3.1, onde aparecem representadas as duas componentes do sinal de saída do sistema variante no tempo para $k = 1$.

Os parâmetros de Markov para o sistema *Benchmark*, para $k = 1$, são:

$$D_1 = \begin{bmatrix} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{bmatrix} \qquad C_1B_1 = \begin{bmatrix} 1.0000 & 4.0000 \\ 0.0000 & 3.0000 \end{bmatrix}$$

$$C_1A_1B_1 = \begin{bmatrix} -9.5007 & -0.5001 \\ -6.5007 & -0.5001 \end{bmatrix}$$

e assim sucessivamente.

Os parâmetros de Markov para o modelo computacionalmente obtido para $k = 1$, são:

$$Dc_1 = \begin{bmatrix} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{bmatrix} \qquad Cc_1Bc_1 = \begin{bmatrix} 0.9998 & 4.0000 \\ -0.0006 & 3.0000 \end{bmatrix}$$

$$Cc_1Ac_1Bc_1 = \begin{bmatrix} -9.5005 & -0.5001 \\ -6.5001 & -0.5002 \end{bmatrix}$$

e assim sucessivamente.

Como os parâmetros de Markov para o sistema *Benchmark*

$$\left[\; D_1, \;\; C_1B_1, \;\; C_1A_1B_1, \;\; C_1A_1^2B_1, \;\; \ldots \; \right]$$

e os parâmetros de Markov do sistema identificado

$$\left[\; Dc_1, \;\; Cc_1Bc_1, \;\; Cc_1Ac_1Bc_1, \;\; Cc_1Ac_1^2Bc_1, \;\; \ldots \; \right]$$

coincidem, concluimos que para $k = 1$ a qualidade do método proposto é adequada.

Da mesma forma, são analisados os parâmetros de Markov dos resultados obtidos para diversos valores de k, nos permitindo à mesma conclusão sobre a adequação do método proposto, por brevidade estes resultados não são aqui apresentados.

No Anexo 8, seção 8.4, as Figs. 8.1, 8.2 e 8.3 apresentam, respectivamente, os sinais de saída do sistema variante no tempo para $k = 4$, $k = 7$ e $k = 10$.

Exemplo 2: Neste caso, escolhemos de 10 valores de instantes de amostragem k_j, $j = 1, .., 10$, apresentados no vetor G:

$$G = \left[\begin{array}{cccccccccc} -2.00 & 1.55 & 5.11 & 8.66 & 12.22 & 15.77 & 19.33 & 22.88 & 26.44 & 30.00 \end{array} \right] \tag{3.69}$$

Através da modelagem computacional utilizando o programa **MOESP_VAR** para um conjunto de 100 amostras de valores entrada-saída, obtivemos o seguinte modelo no espaço de estado, para o primeiro intervalo de experimentação $\left[\begin{array}{cc} -6 & 2 \end{array} \right]$ no ponto $G(1) = -2$:

$$Ac_{-2} = \left[\begin{array}{cc} -0.7298 & 1.2604 \\ -0.2339 & -0.2770 \end{array} \right]; \qquad Bc_{-2} = \left[\begin{array}{cc} -8.5093 & -4.0964 \\ 7.7682 & -2.9518 \end{array} \right]$$

$$Cc_{-2} = \left[\begin{array}{cc} -0.5976 & -0.5258 \\ -0.4093 & -0.4483 \end{array} \right] \qquad Dc_{-2} = \left[\begin{array}{cc} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{array} \right].$$

Para o próximo intervalo: $\left[\begin{array}{cc} -2.44 & 5.55 \end{array} \right]$, obtemos

$$Ac_{1.55} = \left[\begin{array}{cc} -0.7217 & 1.2571 \\ -0.2431 & -0.2732 \end{array} \right]; \qquad Bc_{1.55} = \left[\begin{array}{cc} -8.5527 & -4.1007 \\ 7.8059 & -2.9478 \end{array} \right]$$

$$Cc_{1.55} = \left[\begin{array}{cc} -0.5972 & -0.5262 \\ -0.4092 & -0.4484 \end{array} \right] \qquad Dc_{1.55} = \left[\begin{array}{cc} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{array} \right].$$

Os resultados podem ser observados na Fig. 3.2, onde aparecem representadas, as duas componentes do sinal de saída do sistema variante no tempo, no primeiro e no segundo intervalos de experimentação, $G(1) = -2$ e $G(2) = 1.55$ respectivamente.

Para o próximo intervalo: $\left[\begin{array}{cc} 1.11 & 9.11 \end{array} \right]$, obtemos

$$Ac_{5.11} = \left[\begin{array}{cc} -0.7133 & 1.2547 \\ -0.2529 & -0.2702 \end{array} \right]; \qquad Bc_{5.11} = \left[\begin{array}{cc} -8.5972 & -4.1060 \\ 7.8420 & -2.9430 \end{array} \right]$$

$$Cc_{5.11} = \left[\begin{array}{cc} -0.5967 & -0.5267 \\ -0.4091 & -0.4485 \end{array} \right] \qquad Dc_{5.11} = \left[\begin{array}{cc} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{array} \right].$$

Analogamente para o intervalo $\left[\begin{array}{cc} 4.66 & 12.66 \end{array} \right]$:

$$Ac_{8.66} = \left[\begin{array}{cc} -0.7046 & 1.2532 \\ -0.2633 & -0.2681 \end{array} \right]; \qquad Bc_{8.6667} = \left[\begin{array}{cc} -8.6432 & -4.1121 \\ 7.8760 & -2.9375 \end{array} \right]$$

$$Cc_{8.6667} = \left[\begin{array}{cc} -0.5961 & -0.5272 \\ -0.4090 & -0.4488 \end{array} \right] \qquad Dc_{8.6667} = \left[\begin{array}{cc} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{array} \right].$$

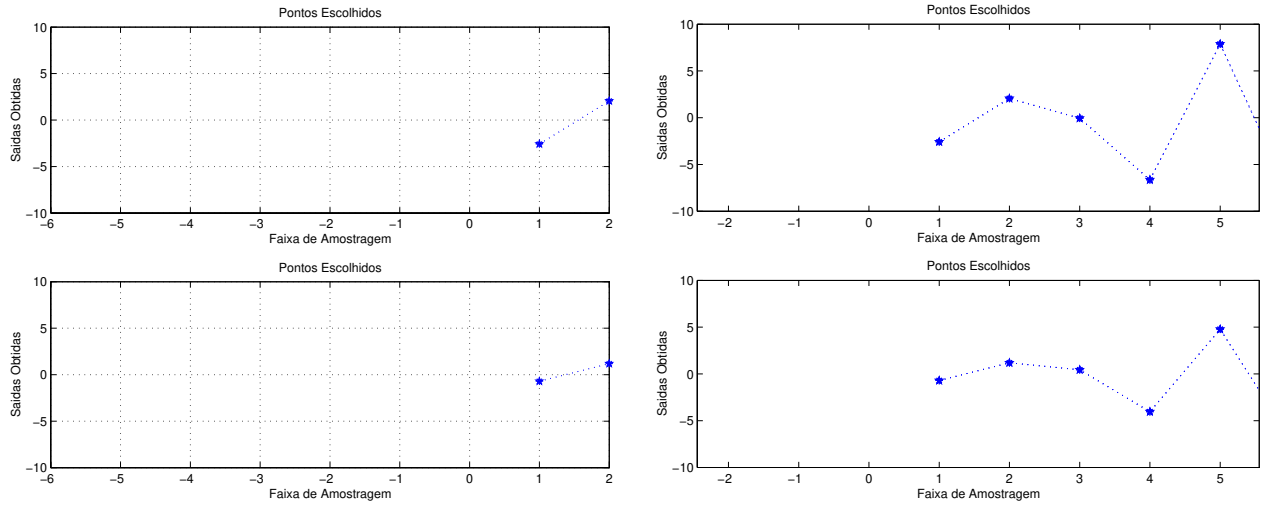

Fig. 3.2: Sinal de Saída para dois Intervalos de Identificação

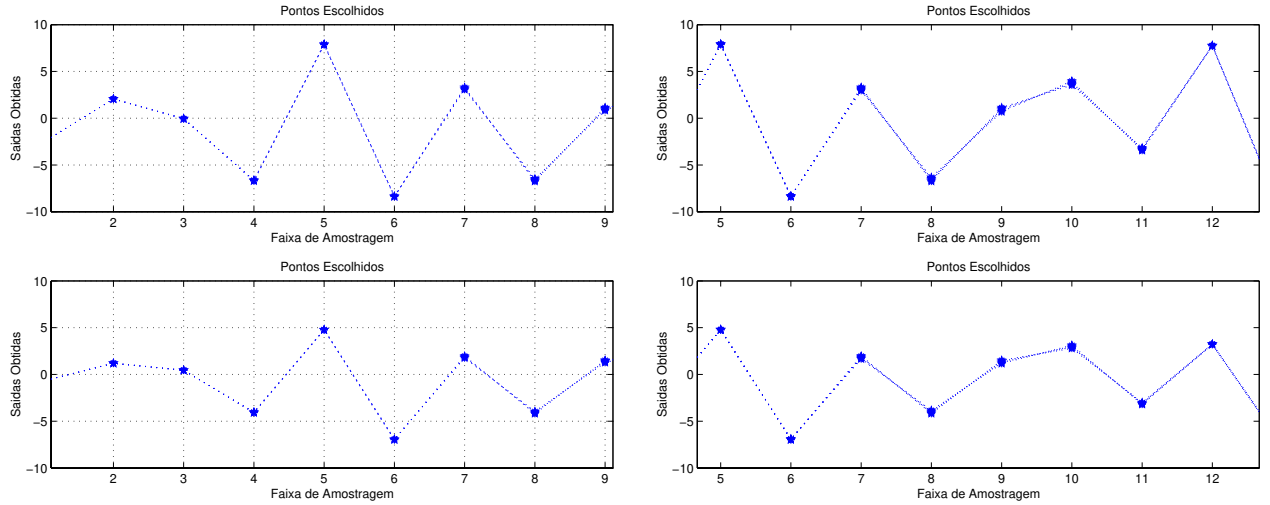

Fig. 3.3: Sinal de Saída para dois Intervalos de Identificação

Na Fig. 3.3, aparecem representadas as duas componentes do sinal de saída do sistema variante no tempo no terceiro e no quarto intervalos de experimentação, $G(3) = 5.11$ e $G(4) = 8.66$ respectivamente.

Analogamente para o intervalo $\begin{bmatrix} 8.22 & 16.22 \end{bmatrix}$:

$$Ac_{12.22} = \begin{bmatrix} -0.6957 & 1.2526 \\ -0.2742 & -0.2669 \end{bmatrix}; \qquad Bc_{12.22} = \begin{bmatrix} -8.6910 & -4.1190 \\ 7.9078 & -2.9315 \end{bmatrix}$$

$$Cc_{12.22} = \begin{bmatrix} -0.5954 & -0.5279 \\ -0.4087 & -0.4491 \end{bmatrix} \qquad Dc_{12.22} = \begin{bmatrix} 1.0000 & 3.0000 \\ 1.0000 & 1.0000 \end{bmatrix}.$$

Nas Figs. 3.4 e 3.5, aparecem representadas as duas componentes do sinal de saída do sistema variante no tempo no quinto e no sexto intervalos de experimentação, $G(5) = 12.22$ e $G(6) = 15.77$ respectivamente.

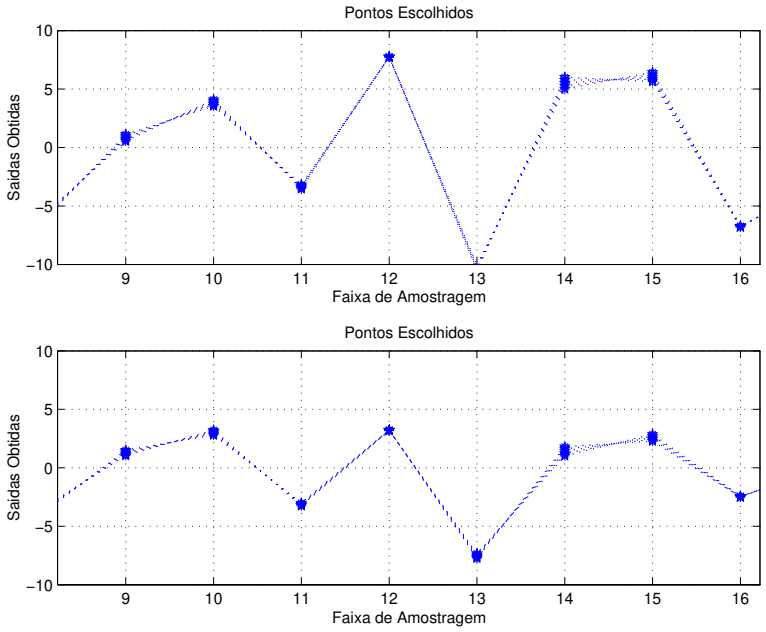

Fig. 3.4: Sinal de Saída para o quinto Intervalo de Identificação

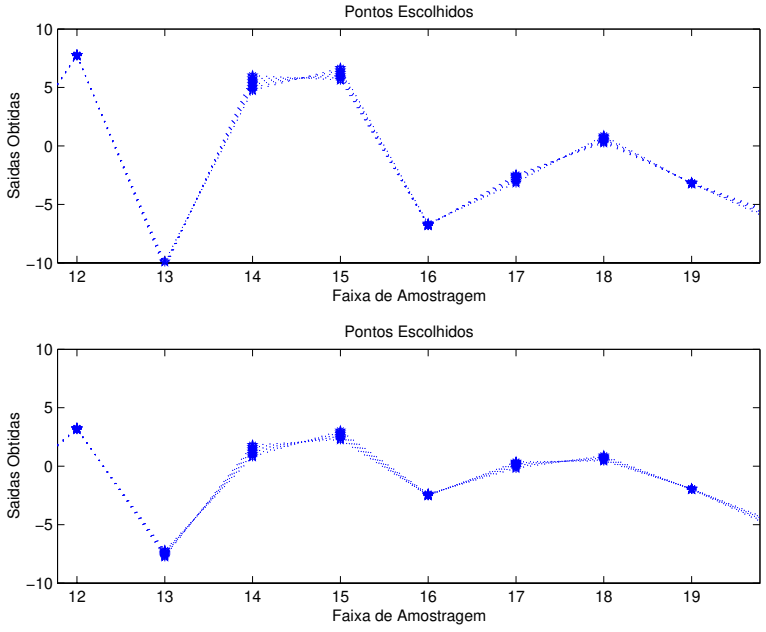

Fig. 3.5: Sinal de Saída para o sexto Intervalo de Identificação

Estes dois exemplos apresentam aspectos da identificação de sistemas variantes no tempo com o algoritmo proposto e evidenciam sua adequação bem como sua complexidade dada nossa falta de intuição e de critérios de desempenho para analisar tais sistemas e seus respectivos dados.

3.7 Modelagem Determinística-Estocástica no Espaço de Estado

Para identificação linear de sistemas discretos multivariáveis ruidosos com entradas exógenas no espaço de estado uma forma de representação é dada por:

$$\left\{ \begin{array}{lcl} x_{k+1} & = & Ax_k + Bu_k + v_k \\ y_k & = & Cx_k + Du_k + w_k \end{array} \right. \tag{3.70}$$

com

$$E\left[\left(\begin{array}{c} v_k \\ w_k \end{array} \right) \left(\begin{array}{cc} v_s^T & w_s^T \end{array} \right) \right] = \left\{ \begin{array}{cl} \left[\begin{array}{cc} Q & S \\ S^T & R \end{array} \right] & k = s \\ 0 & k \neq s \end{array} \right. \tag{3.71}$$

onde $u_k \in \Re^m$, $y_k \in \Re^l$ e $x_k \in \Re^n$ representam o vetor de entrada, o vetor de saída e o vetor de estado respectivamente, no k-ésimo instante de tempo. Sabemos que somente podemos fazer uso dos dados de entrada $(u_k, u_{k+1}, \ldots)$ e de saída $(y_k, y_{k+1}, \ldots)$. Como as matrizes $A_{n\times n}, B_{n\times m}, C_{l\times n}, D_{l\times m}$ não são conhecidas é um objetivo deste trabalho propor e discutir procedimentos numéricos eficientes para encontrá-las, tanto para o caso de sistemas invariantes no tempo, como para o caso de sistemas variantes no tempo, onde estas matrizes podem no todo ou em parte ser variáveis no tempo; w_k e v_k são sinais de ruído nas medidas da saída e no processo.

Segundo [40, 158, 159], o modelo linear (3.70) para o sinal ruidoso y_k pode ser decomposto em um subsistema determinístico, sobrescrito d, e em um subsistema estocástico, sobrescrito e pela decomposição do estado x_k e da saída y_k em componentes determinísticas e estocásticas respectivamente representadas por:

$$x_k = x_k^d + x_k^e \tag{3.72}$$

$$y_k = y_k^d + y_k^e \tag{3.73}$$

O estado determinístico x_k^d e a saída determínistica y_k^d estão associados ao subsistema determinístico que descreve a influência da entrada determínistica u_k sobre a saída determínistica, enquanto que o estado estocástico x_k^e e a saída estocástica y_k^e estão associados ao subsistema estocástico, que descreve a influência das seqüências ruidosas v_k e w_k sobre a saída estocástica:

$$\left\{ \begin{array}{lcl} x_{k+1}^d & = & Ax_k^d + Bu_k \\ y_k^d & = & Cx_k^d + Du_k \end{array} \right. \tag{3.74}$$

$$\left\{ \begin{array}{ll} x_{k+1}^e & = Ax_k^e + v_k \\ y_k^e & = Cx_k^e + w_k \end{array} \right. \tag{3.75}$$

respectivamente.

Observamos que nesta decomposição A e C são as mesmas para os dois subsistemas propostos por [40, 158, 159]. Apesar desta decomposição ser possível, consideramos mais adequada e interessante aos nossos propósitos, a decomposição que propomos no Teorema 1 da seção 3.8, onde o estado do sinal ruidoso será uma composição dos estados x_k^d e x_k^e:

$$x_k = \left[\begin{array}{c} x_k^d \\ x_k^e \end{array} \right] \tag{3.76}$$

em lugar do proposto em (3.72).

3.8 Proposta de Algoritmo para Modelagem Computacional de Dados Ruidosos Multivariáveis

Para identificação linear de sistemas multivariáveis ruidosos com entradas exógenas, ou dito de outra forma, para a modelagem linear de dados de sistemas e de séries temporais multivariáveis, pode-se usar:

1. Técnica da Variável Instrumental, [139].
2. Técnica dos Mínimos Quadrados Totais, [157].
3. Princípio da Superposição, com decomposição do modelo em submodelos a serem superpostos.

Com base na análise dos fundamentos essenciais para modelagem computacional de dados no espaço de estado, e na alternativa 3, propomos e implementamos um algoritmo para Identificação Linear Multivariável no Espaço de Estado de Sistemas Ruidosos Invariantes no Tempo que chamaremos de algoritmo **MOESP_AOKI** ou também de algoritmo **AVW**, em homenagem aos pesquisadores Masanao Aoki, Michel Verhaegen e Patrick Dewilde, que desenvolveram respectivamente o método de *AOKI* para modelagem de Séries Temporais no espaço de estado e o método *MOESP* para identificação multivariável no espaço de estado.

Partindo dos estudos e das implementações do algoritmo de *Aoki* para modelagem de séries temporais multivariáveis no espaço de estado e do algoritmo **MOESP** para a identificação de sistemas no espaço de estado, realizados em [12] e também tratados em [32], neste estudo propomos a criação do algoritmo determinístico-estocástico **AVW** para modelagem computacional de dados ruidosos multivariáveis que combina estas duas excelentes abordagens.

Definimos uma seqüência de instruções, apoiados no estudo de tais algoritmos tratados nesta e nas teses [12, 32] realizadas no LCSI, para modelagem computacional de dados no espaço de estado. Para maiores detalhes, as teses citadas, além dos trabalhos de Aoki [4] e Verhaegen [162] devem ser consultados.

Seja o modelo discreto multivariável

$$\left\{ \begin{array}{lcl} x_{k+1} & = & Ax_k + Bu_k + Fv_k \\ y_k & = & Cx_k + Du_k + Gv_k + w_k \end{array} \right. \tag{3.77}$$

a ser obtido a partir de i amostras de u_k e y_k disponíveis. As entradas ruidosas v_k e w_k são processos estocásticos ruído branco com média zero.

Supomos que:

- A covariância $E\left[\begin{pmatrix} v_{k_1} \\ w_{k_1} \end{pmatrix} \begin{pmatrix} v_{k_2}^T & w_{k_2}^T \end{pmatrix} \right] = \left\{ \begin{array}{ll} \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} & k_1 = k_2 \\ 0 & k_1 \neq k_2 \end{array} \right.$
- O sistema é assintoticamente estável
- $\begin{pmatrix} A & \begin{bmatrix} B & FQ^{1/2} \end{bmatrix} \end{pmatrix}$ é controlável

- $\begin{pmatrix} A & C \end{pmatrix}$ é observável.

Montando as matrizes de Hankel dos dados de saída $Y_H(k)$ com li linhas e T colunas, de entrada $U_H(k)$, com $mi \times T$, e dos ruídos $V_H(k)$, com $li \times T$ e $W_H(k)$ compatível, obtém-se o seguinte modelo estendido ou equação de dados no espaço de estado:

$$Y_H(k) = \mathcal{O}_i \mathcal{X}_H(k) + \mathcal{T}_1 U_H(k) + \mathcal{T}_2 V_H(k) + W_H(k) \tag{3.78}$$

onde

$$Y_H(k) = \begin{bmatrix} y_k & y_{k+1} & \cdots & y_{k+T-1} \\ y_{k+1} & y_{k+2} & \cdots & y_{k+T} \\ \vdots & \vdots & & \vdots \\ y_{k+i-1} & y_{k+i} & \cdots & y_{k+i+T-2} \end{bmatrix} = \begin{bmatrix} y_k^+ & y_{k+1}^+ & \cdots & y_{k+T-1}^+ \end{bmatrix} \tag{3.79}$$

$$U_H(k) = \begin{bmatrix} u_k & u_{k+1} & \cdots & u_{k+T-1} \\ u_{k+1} & u_{k+2} & \cdots & u_{k+T} \\ \vdots & \vdots & & \vdots \\ u_{k+i-1} & u_{k+i} & \cdots & u_{k+i+T-2} \end{bmatrix} = \begin{bmatrix} u_k^+ & u_{k+1}^+ & \cdots & u_{k+T-1}^+ \end{bmatrix} \tag{3.80}$$

$$V_H(k) = \begin{bmatrix} v_k & v_{k+1} & \cdots & v_{k+T-1} \\ v_{k+1} & v_{k+2} & \cdots & v_{k+T} \\ \vdots & \vdots & & \vdots \\ v_{k+i-1} & v_{k+i} & \cdots & v_{k+i+T-2} \end{bmatrix} = \begin{bmatrix} v_k^+ & v_{k+1}^+ & \cdots & v_{k+T-1}^+ \end{bmatrix} \tag{3.81}$$

$\mathcal{X}(k)$ é uma matriz $n \times T$ que contém a seqüência de vetores de estado:

$$\mathcal{X}(k) = \begin{bmatrix} x_k & x_{k+1} & \cdots & x_{k+T-1} \end{bmatrix} \tag{3.82}$$

$\mathcal{T}_1$ é uma matriz Toeplitz triangular inferior $(li) \times (mi)$ contendo os parâmetros de Markov do sistema

$$\mathcal{T}_1 = \begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & & 0 \\ CAB & CB & & 0 \\ \vdots & & & 0 \\ CA^{(i-2)}B & CA^{(i-3)}B & \cdots & D \end{bmatrix} \tag{3.83}$$

Analogamente $\mathcal{T}_2$ é uma matriz Toeplitz:

$$\mathcal{T}_2 = \begin{bmatrix} G & 0 & \cdots & 0 \\ CF & G & & 0 \\ CAF & CF & & 0 \\ \vdots & & & 0 \\ CA^{(i-2)}F & CA^{(i-3)}F & \cdots & G \end{bmatrix} \tag{3.84}$$

e $\mathcal{O}_i$, $(li) \times n$ é a matriz de observabilidade estendida:

$$\mathcal{O}_i = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{(i-1)} \end{bmatrix} \tag{3.85}$$

Fazendo a projeção ortogonal de cada termo do modelo estendido no complemento ortogonal de U_H obtém-se:

$$Y_H(k)|U_H(k)^{\perp} = \mathcal{O}_i\mathcal{X}(k)|U_H(k)^{\perp} + \mathcal{T}_1U_H(k)|U_H(k)^{\perp} + \mathcal{T}_2V_H(k)|U_H(k)^{\perp} + W_H(k)|U_H(k)^{\perp}$$

Pode-se mostrar que algoritmos baseados em projeção ortogonal falham em prover estimativas consistentes de $\mathcal{O}_i$, quando v e w são não nulos. O algoritmo MOESP ordinário é capaz de fornecer estimativas não-polarizadas das matrizes se o sistema for afetado apenas por erro de medida ruído branco na saída (espacial ou temporalmente), [161].

Para o caso de representações mais gerais de ruído, algoritmos baseados em variáveis instrumentais como o PI-MOESP e o PO-MOESP foram propostos em [163]. Nestes esquemas, os conjuntos de dados são subdivididos em parte passada e parte futura. No algoritmo PI as entradas passadas são usadas como variáveis instrumentais, enquanto que no algoritmo PO tanto as entradas passadas como as saídas são usadas. Assim, o algoritmo PI-MOESP identifica a parte do sistema excitada por u, enquanto que o algoritmo PO-MOESP também estima a parte do sistema excitada pelo ruído.

Para o algoritmo PI-MOESP em [163] é demonstrado que, pelo uso da fatoração QR:

$$\begin{bmatrix} U_H(k+i) \\ U_H(k) \\ Y_H(k+i) \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{3.86}$$

sob condições adequadas de excitação persistente aplica-se:

$$\lim_{T\longrightarrow\infty} \frac{1}{\sqrt{T}} Y_H(k+i)Q_2^T = \lim_{T\longrightarrow\infty} \frac{1}{\sqrt{T}} \Gamma\mathcal{X}(k+i)Q_2^T$$

o que permite estimar o subespaço de observabilidade do sistema diretamente dos dados entrada-saída.

De forma similar, para o PO-MOESP, pelo uso da fatoração

$$\begin{bmatrix} U_H(k+i) \\ U_H(k) \\ Y_H(k) \\ Y_H(k+i) \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 & 0 \\ R_{21} & R_{22} & 0 & 0 \\ R_{31} & R_{32} & R_{33} & 0 \\ R_{41} & R_{42} & R_{43} & R_{44} \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{bmatrix} \tag{3.87}$$

obtém-se que

$$\lim_{T\longrightarrow\infty} \frac{1}{\sqrt{T}} Y_H(k+i)Q_2^T = \lim_{T\longrightarrow\infty} \frac{1}{\sqrt{T}} \Gamma\mathcal{X}(k+i)Q_2^T$$

e

$$\lim_{T\longrightarrow\infty} \frac{1}{\sqrt{T}} Y_H(k+i)Q_3^T = \lim_{T\longrightarrow\infty} \frac{1}{\sqrt{T}} \Gamma\mathcal{X}(k+i)Q_3^T$$

tal que também para este caso estimativas não-polarizadas do subespaço de interesse podem ser obtidas, e computadas estimativas das matrizes A e C de forma similar à do algoritmo MOESP ordinário.

Pelas razões apresentadas nesta seção estamos propondo o algoritmo **MOESP_AOKI**, que envolve basicamente o procedimento apresentado a seguir, e que se fundamenta no seguinte Teorema que também propomos:

Teorema 1 *Por superposição, o modelo linear (3.77) para o sinal ruidoso y_k pode ser decomposto nos dois submodelos seguintes:*

$$\left\{ \begin{array}{rcl} x_{k+1}^d & = & A^d x_k^d + B^d u_k \\ y_k^d & = & C^d x_k^d + D^d u_k \end{array} \right. \tag{3.88}$$

$$\left\{ \begin{array}{l} x_{k+1}^e = A^e x_k^e + F^e v_k \\ y_k^e = C^e x_k^e + G^e v_k + w_k \end{array} \right. \tag{3.89}$$

onde o sobrescrito d refere-se a determinístico e o sobrescrito e refere-se a estocástico e $y_k = y_k^d + y_k^e$. O estado do sinal ruidoso é

$$x_k = \left[\begin{array}{c} x_k^d \\ x_k^e \end{array} \right]$$

, bem como

$$A = \left[\begin{array}{cc} A^d & 0 \\ 0 & A^e \end{array} \right] \quad B = \left[\begin{array}{c} B^d \\ 0 \end{array} \right] \quad C = \left[\begin{array}{cc} C^d & C^e \end{array} \right] \quad F = \left[\begin{array}{c} 0 \\ F^e \end{array} \right] \quad D = D^d, G = G^e$$

Prova 1 *Aplicando o princípio da superposição pode-se escrever o modelo linear (3.77) na forma:*

$$y_k = y_k^d + y_k^e = C^d x_k^d + C^e x_k^e + D^d u_k + G^e v_k + w_k$$

com

$$\begin{array}{l} y_k^d = C^d x_k^d + D^d u_k \\ y_k^e = C^e x_k^e + G^e v_k + w_k \end{array}$$

donde

$$y_k = y_k^d + y_k^e = \left[\begin{array}{cc} C^d & C^e \end{array} \right] \left[\begin{array}{c} x_k^d \\ x_k^e \end{array} \right] + D^d u_k + G^e v_k + w_k$$

Como o estado do sinal ruidoso é

$$x_k = \left[\begin{array}{c} x_k^d \\ x_k^e \end{array} \right],$$

então

$$x_{k+1} = \left[\begin{array}{c} x_{k+1}^d \\ x_{k+1}^e \end{array} \right] = \left[\begin{array}{c} A^d x_k^d + B^d u_k \\ A^e x_k^e + F^e v_k \end{array} \right] = \left[\begin{array}{cc} A^d & 0 \\ 0 & A^e \end{array} \right] \left[\begin{array}{c} x_k^d \\ x_k^e \end{array} \right] + \left[\begin{array}{c} B^d \\ 0 \end{array} \right] u_k + \left[\begin{array}{c} 0 \\ F^e \end{array} \right] v_k$$

donde resulta o sistema (3.77).

Algoritmo MOESP_AOKI

Supondo o modelo sujeito a ruído na forma inovativa, onde e_k é a inovação:

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k + Ke_k \\ y_k &= Cx_k + Du_k + e_k \end{cases} \tag{3.90}$$

a partir dos dados obtidos experimentalmente para um sistema multivariável em ambiente ruidoso e excitado pela entrada u_k, o procedimento computacional é dividido em duas partes. Na primeira parte do algoritmo *AVW* é feita a modelagem computacional determinística apoiada no algoritmo *MOESP* bem como nos desenvolvimentos feitos no Capítulo 2 e 3 relacionados com modelagem determinística no espaço de estado. Na segunda parte do algoritmo *AVW*, é feita a modelagem computacional estocástica, apoiada no algoritmo de *Aoki* bem como nos desenvolvimentos feitos no Capítulo 2 relacionados com modelagem estocástica no espaço de estado, e nos seguintes comentários importantes sobre realização estocástica.

No Capítulo 2 consideramos o estudo de um processo estocástico y_k representado pelo mínimo sistema no espaço de estado de dimensão n dado pelo modelo inovativo:

$$\begin{cases} x_{k+1} &= \mathbf{A}x_k + Ke_k \\ y_k &= \mathbf{C}x_k + e_k \end{cases} \tag{3.91}$$

onde e_k é uma seqüência ruído branco com matriz de covariância dada por $\Delta = E(e_k e_k^T)$.

Na seção 2.3.1 apresentamos como a matriz de covariância para a saída $\{y_k\}$ do processo inovativo pode ser expressa em termos da matriz resposta ao impulso e das matrizes do modelo inovativo, (2.53-2.54) ou equivalentemente:

$$\Lambda_0 = E(y_k y_k^T) = \sum_{k=0}^{\infty} (CA^{k-1}K)\Delta(CA^{k-1}K)^T \tag{3.92}$$

e

$$\Lambda_i = E(y_{k+i} y_k^T) = \sum_{k=0}^{\infty} (CA^{k+i-1}K)\Delta(CA^{k+i-1}K)^T \tag{3.93}$$

Supondo que um conjunto de dados de saída está disponível:

$$y_k \quad y_{k-1} \quad y_{k-1} \quad \cdots \quad y_{k-L+1}$$

A matriz de covariância entre as pilhas de vetores com os dados passados e com os dados futuros da Série Temporal tem a estrutura de uma matriz de Hankel:

$$E[y_{k+1}^+ y_k^{-T}] = E\left[\begin{pmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \\ y_{k+J} \end{pmatrix} \begin{pmatrix} y_k^T & y_{k-1}^T & y_{k-2}^T & \cdots & y_{k-L+1}^T \end{pmatrix}\right] = \begin{bmatrix} \Lambda_1 & \Lambda_2 & \cdots & \Lambda_L \\ \Lambda_2 & \Lambda_3 & \cdots & \Lambda_{L+1} \\ \vdots & \vdots & \vdots & \ddots \\ \Lambda_J & \Lambda_{J+1} & \cdots & \Lambda_{J+L-1} \end{bmatrix} \tag{3.94}$$

onde J refere-se ao horizonte para o futuro.

A partir das equações (3.91) tem-se que $\Pi = E[x_k x_k^T]$ satisfaz as relações, vide (2.33,2.50) e (2.49) ou equivalentemente:

$$\Pi = A\Pi A^T + K\Delta K^T$$
$$M = A\Pi C^T + K\Delta$$
$$\Lambda_0 = C\Pi C^T + \Delta$$

Supondo $\Delta > 0$, Δ e K podem ser expressos como:

$$\Delta = \Lambda_0 - C\Pi_0 C^T$$
$$K = (M - A\Pi C^T)\Delta^{-1}$$

A partir destas equações tem-se:

$$\Pi = A\Pi A^T + (M - A\Pi C^T)(\Lambda_0 - C\Pi_0 C^T)^{-1}(M - A\Pi C^T)^T \tag{3.95}$$

Logo, o problema de realização estocástica pode ser decomposto nas seguintes etapas:

1. Determinar as matrizes Λ_0, A, M e C que representam um modelo para uma seqüência de covariâncias Λ_i de um conjunto de saídas y_k. Para calcular Λ_0, A, M e C pode-se utilizar um procedimento semelhante ao utilizado para a realização determinística agora supondo que a matriz de Hankel de covariâncias possa ser fatorada como $H = \mathcal{O}\Omega$ com $\Omega = (M, AM, ...)$.

2. Resolver a equação de Riccati, (3.95), para obter a covariância do estado Π.

3. Calcular Δ e K a partir de Λ_0, A, M, C e Π.

Portanto, o procedimento para o algoritmo *MOESP_AOKI* resume-se ao seguinte:

- Na primeira parte do algoritmo *MOESP_AOKI*, aplicar o algoritmo *MOESP*:
  - Cálcular as matrizes U_p, U_f, Y_p, Y_f;

$$U_p = \begin{bmatrix} u_0 & u_1 & u_2 & \dots & u_{N-1} \\ u_1 & u_2 & u_3 & \dots & u_N \\ u_2 & u_3 & u_4 & \dots & u_{N+1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ u_{j-1} & u_j & u_{j+1} & \dots & u_{N+j-2} \end{bmatrix}; \quad U_f = \begin{bmatrix} u_j & u_{j+1} & u_{j+2} & \dots & u_{N+j-1} \\ u_{j+1} & u_{j+2} & u_{j+3} & \dots & u_{N+j} \\ u_{j+2} & u_{j+3} & u_{j+4} & \dots & u_{N+j+1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ u_{2j-1} & u_2 j & u_{2j+1} & \dots & u_{N+2j-2} \end{bmatrix}$$

$$Y_p = \begin{bmatrix} y_0 & y_1 & y_2 & \dots & y_{N-1} \\ y_1 & y_2 & y_3 & \dots & y_N \\ y_2 & y_3 & y_4 & \dots & y_{N+1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ y_{j-1} & y_j & y_{j+1} & \dots & y_{N+j-2} \end{bmatrix}; \quad Y_f = \begin{bmatrix} y_j & y_{j+1} & y_{j+2} & \dots & y_{N+j-1} \\ y_{j+1} & y_{j+2} & y_{j+3} & \dots & y_{N+j} \\ y_{j+2} & y_{j+3} & y_{j+4} & \dots & y_{N+j+1} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ y_{2j-1} & y_2 j & y_{2j+1} & \dots & y_{N+2j-2} \end{bmatrix}$$

  - Obter a quadrúpla de matrizes A^d, B^d, C^d, D^d, bem como y_k^d.

- Determinar o sinal y_k^e, que por simplicidade também denotaremos como $\bar{y}$.

- Na segunda parte do algoritmo *MOESP_AOKI*, aplicar o algoritmo *AOKI* a y_k^e:
  - Gerar as matrizes $H^A, H^M, H^C, H, Y_-, Y_+$;

$$Y_- = \begin{bmatrix} \bar{y}_1 & \bar{y}_2 & \bar{y}_3 & \dots & \bar{y}_{N-1} \\ 0 & \bar{y}_1 & \bar{y}_2 & \dots & \bar{y}_{N-2} \\ 0 & 0 & \bar{y}_1 & \dots & \bar{y}_{N-3} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \bar{y}_{N-k-1} & \bar{y}_{N-k} \end{bmatrix}; \quad Y_+ = \begin{bmatrix} \bar{y}_2 & \bar{y}_3 & \bar{y}_4 & \dots & \bar{y}_N \\ \bar{y}_3 & \bar{y}_4 & \bar{y}_5 & \dots & 0 \\ \bar{y}_4 & \bar{y}_5 & \bar{y}_6 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \bar{y}_{j+1} & \bar{y}_{j+2} & \bar{y}_{j+3} & \dots & 0 \end{bmatrix}$$

$$H = \frac{Y_+ Y_-^T}{N} = \begin{bmatrix} \Lambda_1 & \Lambda_2 & \dots & \Lambda_k \\ \Lambda_2 & \Lambda_3 & \dots & \Lambda_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_j & \Lambda_{j+1} & \dots & \Lambda_{j+k} \end{bmatrix} \tag{3.96}$$

$$H^A = \begin{bmatrix} \Lambda_2 & \Lambda_3 & \dots & \Lambda_{k+1} \\ \Lambda_3 & \Lambda_4 & \dots & \Lambda_{k+2} \\ \vdots & \vdots & \ddots & \vdots \\ \Lambda_{j+1} & \Lambda_{j+2} & \dots & \Lambda_{j+k+1} \end{bmatrix}; \quad H^M = \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_j \end{bmatrix} \tag{3.97}$$

$$H^C = \begin{bmatrix} \Lambda_1 & \Lambda_2 & \dots & \Lambda_k \end{bmatrix} \tag{3.98}$$

  - Obter a decomposição em valores singulares da matriz de Hankel das covariâncias;

$$H = U \Sigma^{1/2} \Sigma^{1/2} V^T \tag{3.99}$$

  - Cálcular as matrizes A^e, C^e, K^e;
  - Validar.

Para verificar a estrutura das matrices acima, vide [26, 4].

3.9 Exemplos de Modelagem de Dados Ruidosos

3.9.1 Resultados com N4SID

O **Numerical algorithm for Subspace State System IDentification (N4SID)** é um algoritmo de Identificação que estima modelos no espaço de estado usando método de subespaço. Este algoritmo manipula um número arbitrário de entradas e saídas, incluíndo o caso das séries temporais.

Nesta seção fazemos uma comparação do algoritmo **MOESP_AOKI** com o algoritmo de subespaço **N4SID**, apresentado na literatura como um método combinado para identificação de sistemas invariantes no tempo no espaço de estado tanto determinísticos como estocásticos, através de exemplos.

Primeiramente, uma seqüência de entradas-saídas, vide Tabelas 8.11 e 8.10 na seção 8.4 do Anexo, com vetor de entrada multivariada de 3 elementos e vetor de saída multivariada de 2 elementos, é obtida a partir de uma realização *Benchmark* no espaço de estado. Obtemos um conjunto de dados com 100 amostras, com um modelo inovativo da seguinte forma:

$$\begin{cases} x_{k+1} &= \mathbf{A}x_k + \mathbf{B}u_k + \mathbf{K}e_k \\ y_{kr} &= \mathbf{C}x_k + \mathbf{D}u_k + e_k \end{cases} \tag{3.100}$$

com as seguintes matrizes $A \in \mathcal{R}^{4\times4}$; $B \in \mathcal{R}^{4\times3}$; $C \in \mathcal{R}^{2\times4}$; $K \in \mathcal{R}^{4\times2}$ e $D \in \mathcal{R}^{2\times3}$:

$$A = \begin{bmatrix} 0.2128 & 0.1360 & 0.1979 & -0.0836 \\ 0.1808 & 0.4420 & -0.3279 & 0.2344 \\ -0.5182 & 0.1728 & -0.5448 & -0.3083 \\ 0.2252 & -0.0541 & -0.4679 & 0.8290 \end{bmatrix}, \quad B = \begin{bmatrix} -0.0101 & 0.0317 & -0.9347 \\ -0.0600 & 0.5621 & 0.1657 \\ -0.3310 & -0.3712 & -0.5846 \\ -0.2655 & 0.4255 & 0.2204 \end{bmatrix}, \tag{3.101}$$

$$C = \begin{bmatrix} 0.6557 & -0.2502 & -0.5188 & -0.1229 \\ 0.6532 & -0.1583 & -0.0550 & -0.2497 \end{bmatrix}, \quad D = \begin{bmatrix} -0.4326 & 0.1253 & -1.1465 \\ -1.6656 & 0.2877 & 1.1909 \end{bmatrix}, \tag{3.102}$$

$$K = \begin{bmatrix} -0.0016 & 0.2209 \\ -1.6146 & -1.0061 \\ -1.2287 & -0.4531 \\ 0.2074 & 1.3995 \end{bmatrix}, \tag{3.103}$$

O programa **N4SID** também foi utilizado para validação dos programas que fizemos, implementamos e executamos no *LCSI* da FEEC-UNICAMP.

- Neste primeiro exemplo, fazemos identificação determinística empregando o N4SID. Os dados de entrada e saída u_k, y_k podem ser encontrados nas Tabelas 8.12 e 8.13 da seção 8.4 no Anexo. As matrizes do sistema Benchmark são as apresentadas nas expressões (3.101-3.103).

  A Figura 3.6 apresenta os sinais a identificar y_k e identificado y_{kc}; este último corresponde ao sinal identificado deterministicamente. Pode-se observar que as duas curvas coincidem o que significa que o modelo obtido na identificação descreve com exatidão o sinal y_k.

  As matrizes obtidas na identificação utilizando o método N4SID foram:

$$A_c = \begin{bmatrix} -0.1065 & 0.3924 & -0.1439 & -0.5348 \\ -0.1529 & 0.4471 & 0.2976 & -0.1404 \\ 0.1578 & 0.1871 & 0.7087 & 0.1677 \\ -0.6633 & 0.3422 & -0.3103 & -0.1103 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0.0599 & 0.0861 & 0.0022 \\ 0.0064 & -0.0295 & -0.0935 \\ 0.0163 & -0.0343 & 0.0647 \\ -0.0127 & -0.0519 & 0.0411 \end{bmatrix},$$

$$C_c = \begin{bmatrix} 3.9822 & 5.247 & -0.7677 & 3.7276 \\ 0.95763 & 7.1385 & -0.3918 & 0.7508 \end{bmatrix}, \quad D_c = \begin{bmatrix} -0.4326 & 0.1253 & -1.1465 \\ -1.6656 & 0.2877 & 1.1909 \end{bmatrix}$$

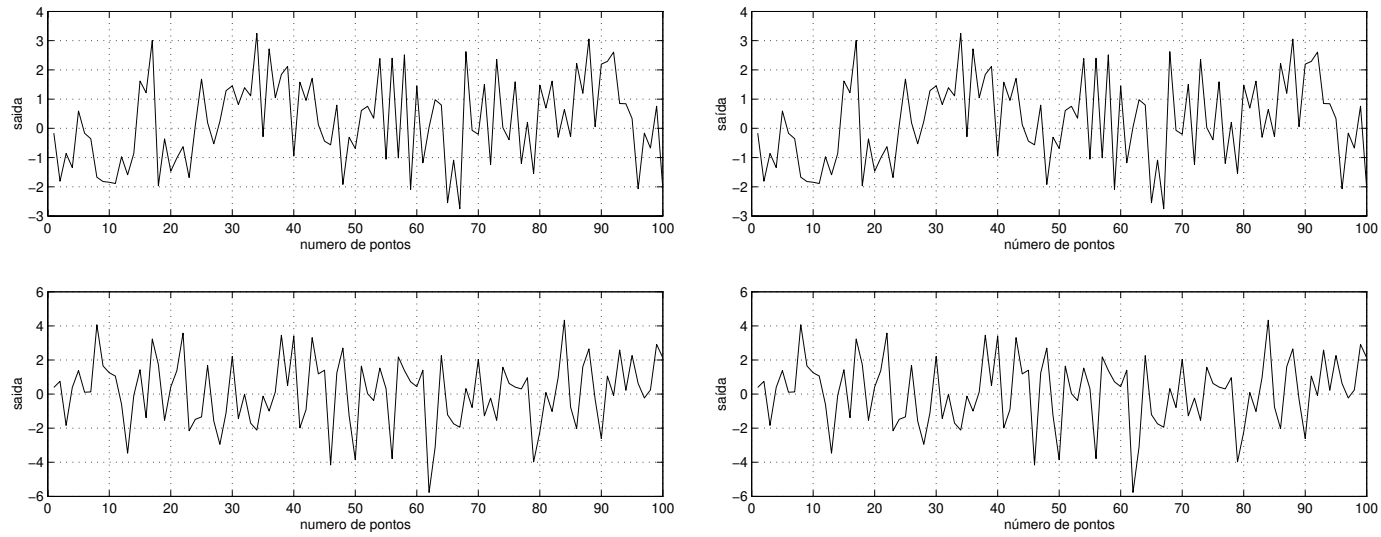

Fig. 3.6: Sinais das saídas y_k, y_{kc}

Pode-se observar que os parâmetros de Markov do Benchmark são:

$$CB = \begin{bmatrix} 0.2127 & 0.0204 & -0.3781 \\ 0.0874 & -0.1541 & -0.6597 \end{bmatrix} \quad CAB = \begin{bmatrix} -0.1651 & -0.2765 & -0.6657 \\ -0.0360 & -0.2262 & -0.3273 \end{bmatrix}$$

e assim sucessivamente, enquanto que os parâmetros de Markov do sistema calculado são:

$$C_cB_c = \begin{bmatrix} 0.2127 & 0.0204 & -0.3781 \\ 0.0874 & -0.1541 & -0.6597 \end{bmatrix} \quad C_cA_cB_c = \begin{bmatrix} -0.1651 & -0.2765 & -0.6657 \\ -0.0360 & -0.2262 & -0.3273 \end{bmatrix}$$

e assim sucessivamente. Como os parâmetros de Markov do *Benchmark* e do sistema identificado são idênticos, concluímos que o procedimento de identificação do N4SID é adequado para a identificação determinística deste sistema como comprovado na Figura 3.6.

- Neste segundo exemplo, fazemos a identificação determinística-estocástica de sistema sujeito a ruído. O sinal y_{kr} é sabido provir de um ruído colorido e_k e de uma entrada u_k. As matrizes de dados são apresentadas nas Tabelas 8.17 e 8.12 da seção 8.4 no Anexo. Os resultados obtidos na identificação com o método **N4SID** são apresentados a seguir:

$$A_c = \begin{bmatrix} -0.1944 & -0.9030 & 0.3592 & 0.2562 \\ 0.7961 & 0.0355 & 0.4118 & 0.4625 \\ 0.1349 & 0.3030 & -0.4674 & 0.6381 \\ -0.1794 & 0.1155 & 0.0256 & 0.3855 \end{bmatrix}, \quad B_c = \begin{bmatrix} -0.00588 & -0.01372 & -0.03586 \\ 0.03009 & -0.01586 & -0.00486 \\ -0.02964 & -0.02922 & 0.03808 \\ -0.00983 & -0.01624 & -0.01994 \end{bmatrix},$$

$$C_c = \begin{bmatrix} 3.4506 & -1.7569 & -4.8586 & -1.4837 \\ -5.3055 & 0.70354 & -3.0513 & 2.1877 \end{bmatrix}, \quad D_c = \begin{bmatrix} -0.2630 & -0.0032 & -1.1287 \\ -1.8149 & 0.2825 & 1.1345 \end{bmatrix}$$

$$K_c = \begin{bmatrix} -0.020715 & 0.00013406 \\ -0.0086275 & -0.025754 \\ -0.0010309 & -0.0079252 \\ -0.0042048 & 0.017819 \end{bmatrix} \tag{3.104}$$

Na Fig.3.7 apresentamos o sinal y_{kr} do Benchmark e a saída y_{krc} do sistema inovativo identificado.

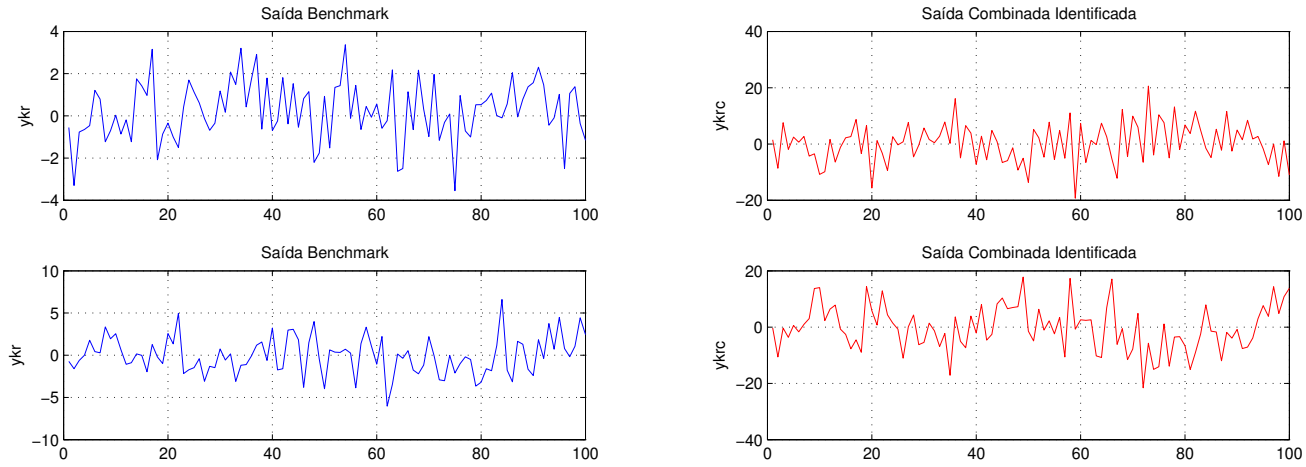

Fig. 3.7: Sinais das saídas y_{kr}, y_{krc}, respectivamente

Na Fig.3.8 apresentamos o erro na modelagem destes dados pelo algoritmo **N4SID**.

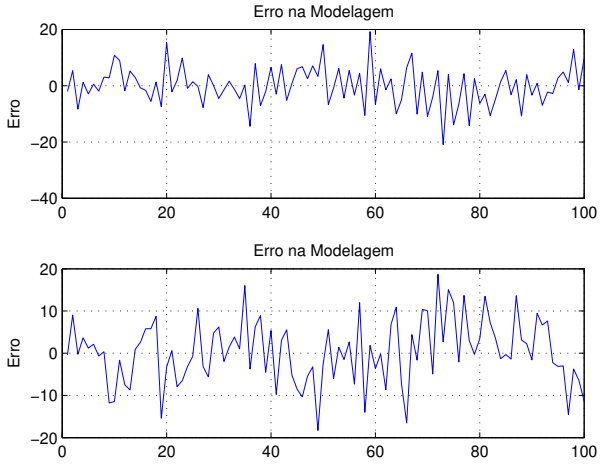

Fig. 3.8: Análise do programa N4SID

Pode-se observar que os parâmetros de Markov do Benchmark são:

$$CB = \begin{bmatrix} 0.2127 & 0.0204 & -0.3781 \\ 0.0874 & -0.1541 & -0.6597 \end{bmatrix} \quad CAB = \begin{bmatrix} -0.1651 & -0.2765 & -0.6657 \\ -0.0360 & -0.2262 & -0.3273 \end{bmatrix}$$

e assim sucessivamente. Como os parâmetros de Markov do sistema calculado são:

$$C_cB_c = \begin{bmatrix} 0.0854 & 0.1466 & -0.2706 \\ 0.1213 & 0.1152 & 0.0270 \end{bmatrix} \quad C_cA_cB_c = \begin{bmatrix} -0.1767 & 0.0884 & 0.2881 \\ 0.1450 & -0.0379 & -0.0108 \end{bmatrix}$$

e assim sucessivamente, concluímos que os dois conjuntos são completamente diferentes e podemos dizer que o procedimento de identificação através do algoritmo N4SID, não foi adequado para a identificação combinada, como foi comprovado na Figura 3.8.

3.9.2 Resultados com MOESP_AOKI

Depois de ter analisado e estudado os resultados experimentais obtidos com a execução do programa **N4SID**, apresentamos nesta seção os resultados obtidos com nossa proposta de algoritmo combinado **AVW** para o mesmo sistema Benchmark.

- Neste terceiro exemplo, fazemos a identificação determinística-estocástica de sistema sujeita a ruído. O sinal y_{kr} em resposta a um ruído branco e_k e a uma entrada u_k apresentada na Tabela 8.12 do Anexo, é apresentado na Tabela 8.16. Os resultados obtidos executando a primeira parte do algoritmo *AVW* são apresentados a seguir:

$$A_c^d = \begin{bmatrix} 0.4815 & 0.2535 & -0.0183 & -0.5331 \\ -0.1405 & -0.6357 & -0.7920 & -0.0680 \\ -0.0095 & 0.5059 & -0.1281 & 0.0126 \\ 0.0703 & -0.1882 & 0.1893 & 0.7128 \end{bmatrix}, \quad B_c^d = \begin{bmatrix} -0.4048 & 0.1314 & 1.0634 \\ 0.3241 & 0.1683 & -0.0379 \\ 0.4276 & -0.0154 & 0.3634 \\ 0.0866 & -0.1299 & 0.0139 \end{bmatrix}$$

$$C_c^d = \begin{bmatrix} -0.5104 & 0.5799 & -0.3539 & 0.0682 \\ -0.6820 & -0.1096 & 0.3748 & -0.3241 \end{bmatrix}, \quad D_c^d = \begin{bmatrix} -0.2248 & 0.0633 & -1.1547 \\ -1.7369 & 0.2979 & 1.2561 \end{bmatrix}$$

Na Fig.3.9 apresentamos a saída do Modelo Determinístico y_k^d.

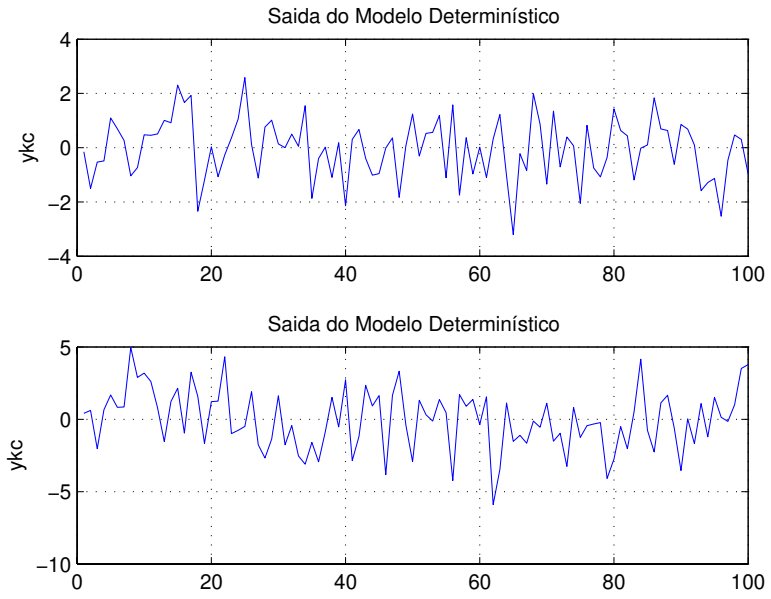

Fig. 3.9: Saída do Modelo Determinístico

A seguir determinamos o sinal que representa um ruído colorido, determinamos uma matriz Hankel $H \in R^{8\times 8}$, e obtemos o seguinte modelo executando a segunda parte do algoritmo **AVW**:

$$\Delta = \begin{bmatrix} 1.3483 & 0.5687 \\ 0.5687 & 1.3177 \end{bmatrix} \qquad A_c^e = \begin{bmatrix} 0.7607 & -0.0674 & -0.0396 & -0.0645 \\ 0.1096 & 0.6169 & 0.6044 & 0.3316 \\ 0.0280 & -0.7476 & 0.4905 & 0.1422 \\ 0.0671 & 0.2146 & -0.1350 & -0.6292 \end{bmatrix}$$

$$K = \begin{bmatrix} -0.5318 & -0.4350 \\ 0.2187 & 0.1866 \\ 0.2310 & 0.0443 \\ 0.3592 & -0.0562 \end{bmatrix} \qquad C_c^e = \begin{bmatrix} -0.9127 & -0.0720 & -0.3361 & 0.1175 \\ -0.9496 & -0.1526 & 0.1805 & -0.3248 \end{bmatrix}$$

Na Fig.3.10 apresentamos o sinal modelado, y_k^e, usando a segunda parte do algoritmo **AVW**. A seguir apresentamos, na Fig.3.11 uma superposição dos sinais y_k^d e y_k^e. Finalmente fazemos uma verificação para validar nossa proposta de algoritmo combinado **AVW** e observamos na Fig.3.12, que para um conjunto de dados entrada-saída, com ruído branco, o algoritmo **AVW** consegue descrever com exatidão o sinal ruidoso y_{kr}.

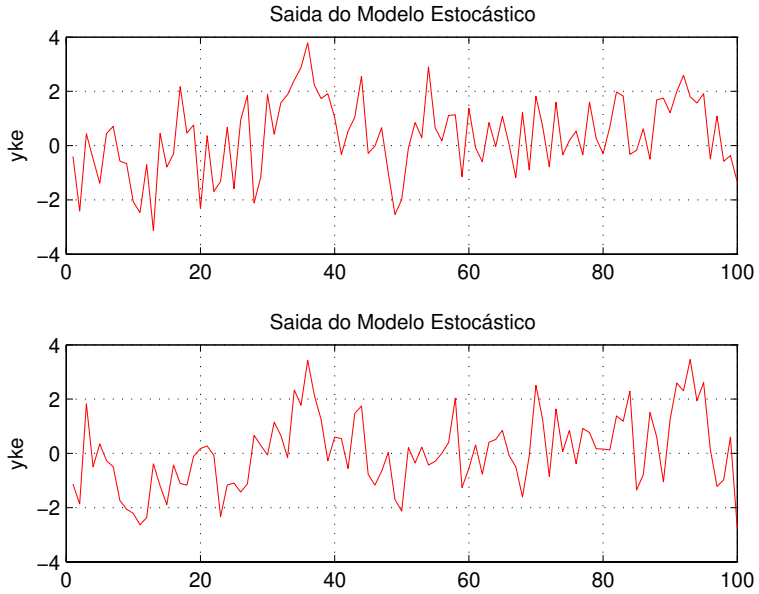

Fig. 3.10: Saída do Modelo Estocástico

- Neste quarto exemplo, também fazemos identificação determinística-estocástica de sinal ruidoso. O sinal y_{kr} em resposta a um ruído colorido e_{kc} e a uma entrada u_k, apresentada na Tabela 8.12 do Anexo, é apresentado na Tabela 8.17. Os resultados obtidos executando a primeira parte

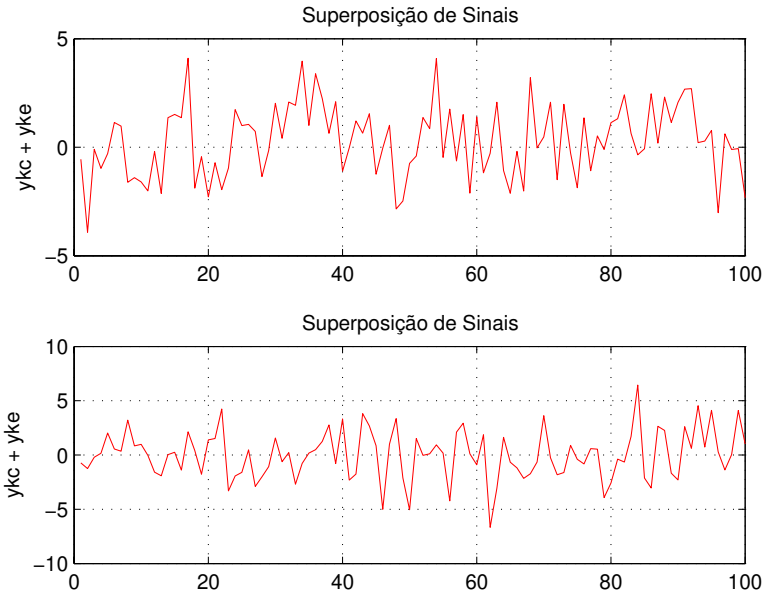

Fig. 3.11: Superposição dos Sinais

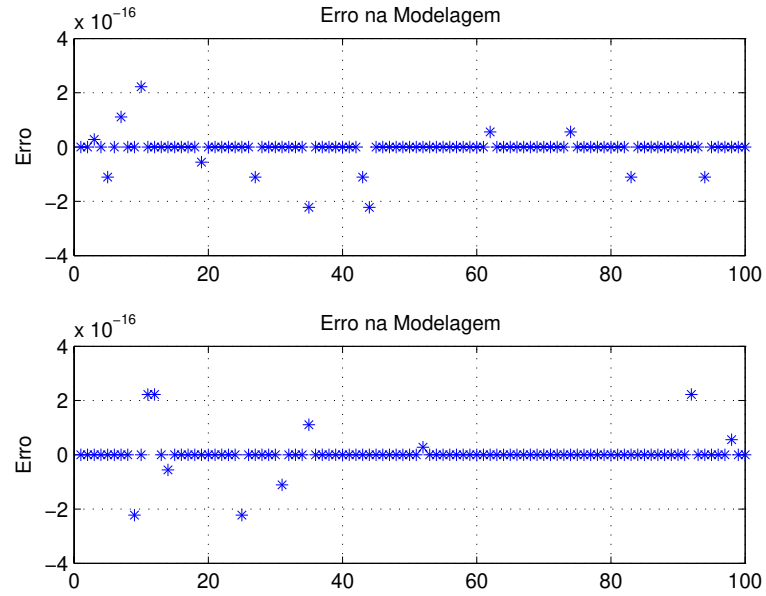

Fig. 3.12: Análise do Programa AVW

do algoritmo **AVW** são apresentados a seguir:

$$A_c^d = \begin{bmatrix} 0.6327 & 0.3401 & -0.2703 & -0.4316 \\ -0.2289 & -0.6774 & -0.5847 & -0.1559 \\ -0.0373 & 0.2491 & 0.0508 & 0.8340 \\ -0.0057 & 0.0857 & -0.2261 & -0.2711 \end{bmatrix}, \quad B_c^d = \begin{bmatrix} -0.5686 & 0.5162 & 2.1037 \\ 0.7688 & 0.4896 & 0.5163 \\ 0.4601 & -0.0252 & 0.4325 \\ 0.1352 & -0.3130 & 0.0065 \end{bmatrix}$$

$$C_c^d = \begin{bmatrix} -0.5143 & 0.5886 & -0.5097 & -0.1458 \\ -0.5417 & 0.1285 & 0.4150 & -0.2605 \end{bmatrix}, \quad D_c^d = \begin{bmatrix} -0.4477 & 0.1460 & -1.1229 \\ -1.7228 & 0.2539 & 1.2776 \end{bmatrix}$$

A Fig.3.13 apresenta a saída do modelo determinístico obtido executando a primeira parte do algoritmo **AVW**.

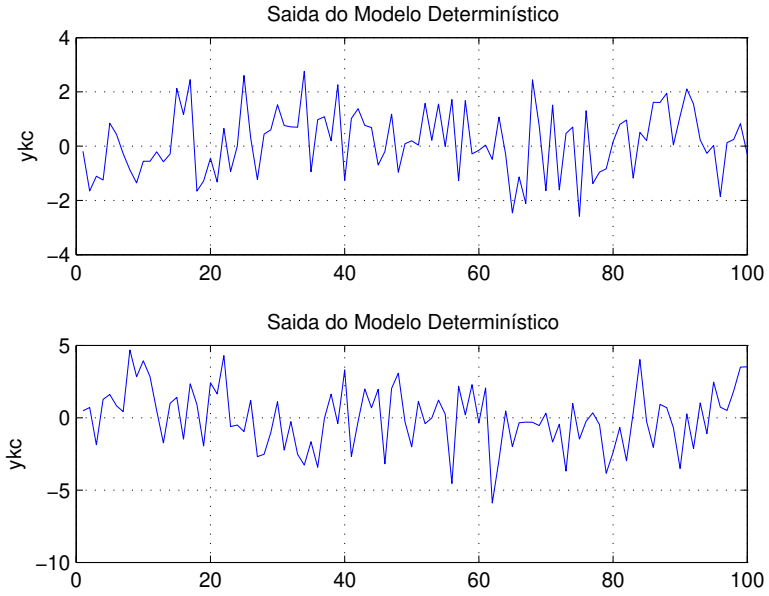

Fig. 3.13: Saída do Modelo Determinístico

A seguir determinamos um sinal que representa um ruído colorido, determinamos uma matriz Hankel $H \in R^{8\times 8}$,e obtemos o seguinte modelo executando a segunda parte do algoritmo **AVW**:

$$\Delta = \begin{bmatrix} 2.7322 & 2.1003 \\ 2.1003 & 3.1580 \end{bmatrix}, \quad A_c^e = \begin{bmatrix} 0.7512 & 0.1753 & 0.0034 & -0.0400 \\ -0.2018 & 0.1127 & 0.1971 & 0.0477 \\ -0.0060 & 0.6146 & 0.2208 & 0.6705 \\ 0.0258 & 0.4929 & -0.3827 & -0.5469 \end{bmatrix}$$

$$K = \begin{bmatrix} -0.7441 & -0.8263 \\ -0.8370 & -0.5541 \\ 0.2580 & 0.0613 \\ 0.0776 & 0.0203 \end{bmatrix} \quad C_c^e = \begin{bmatrix} -1.3274 & 0.6933 & -0.4045 & 0.0388 \\ -1.6072 & 0.2860 & 0.3068 & -0.2560 \end{bmatrix},$$

onde Δ corresponde à matriz de autocovariância do ruído.

Na Fig.3.14 apresentamos o sinal modelado usando a segunda parte do algoritmo **AVW**. A seguir apresentamos uma superposição dos sinais y_{kc}^d e y_{kc}^e. Finalmente fazemos uma verificação para validar nossa proposta de algoritmo combinado **AVW** e observamos na Fig.3.16 que para um conjunto de dados entrada-saída com ruído colorido o algoritmo consegue descrever com exatidão o sinal ruidoso y_{kr}.

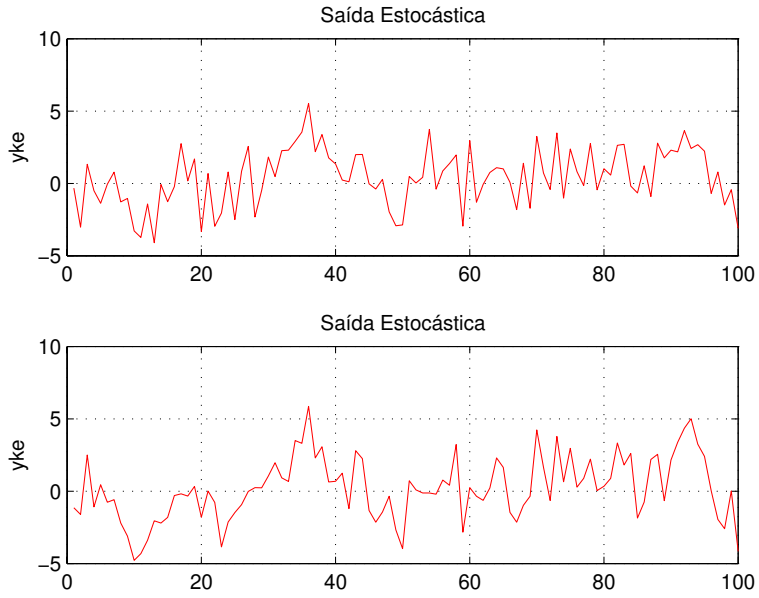

Fig. 3.14: Sinal de saída modelada

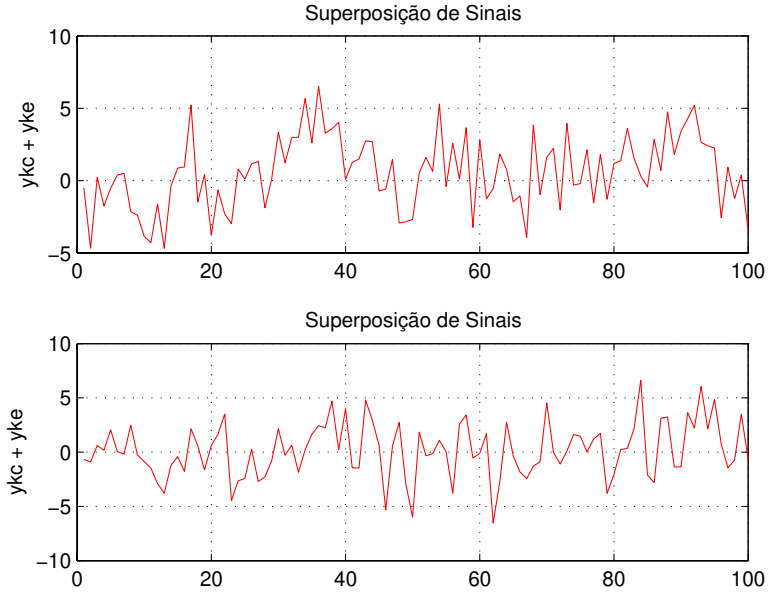

Fig. 3.15: Superposição dos Sinais

Como resultado destes exemplos podemos dizer que nossa proposta de algoritmo **AVW** obtém resultados melhores que o algoritmo **N4SID** quando este é aplicado em forma combinada. Na Fig.3.17 apresentamos as duas saídas obtidas com os algoritmos **N4SID** e **Moesp_Aoki**, respectivamente ao modelar dados provenientes de um modelo considerado sujeito a ruído

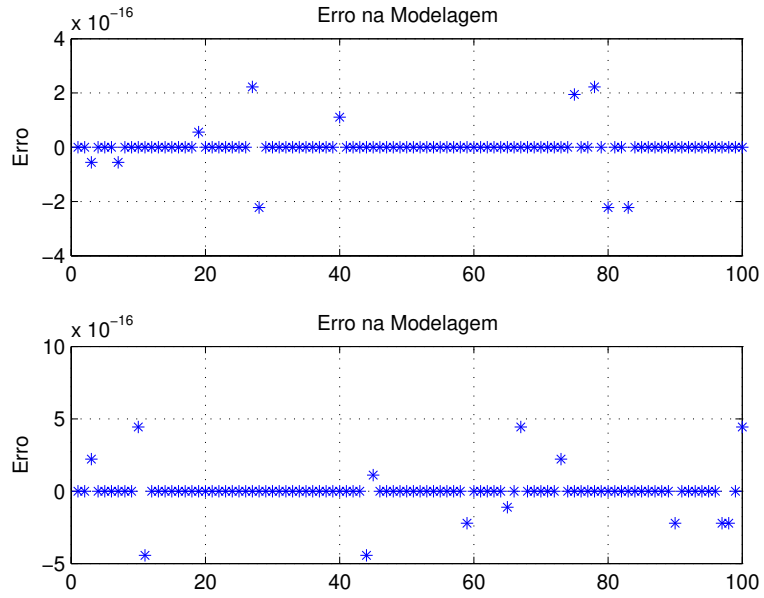

Fig. 3.16: Análise de Erros do Programa **AVW**

Benchmark no espaço de estado.

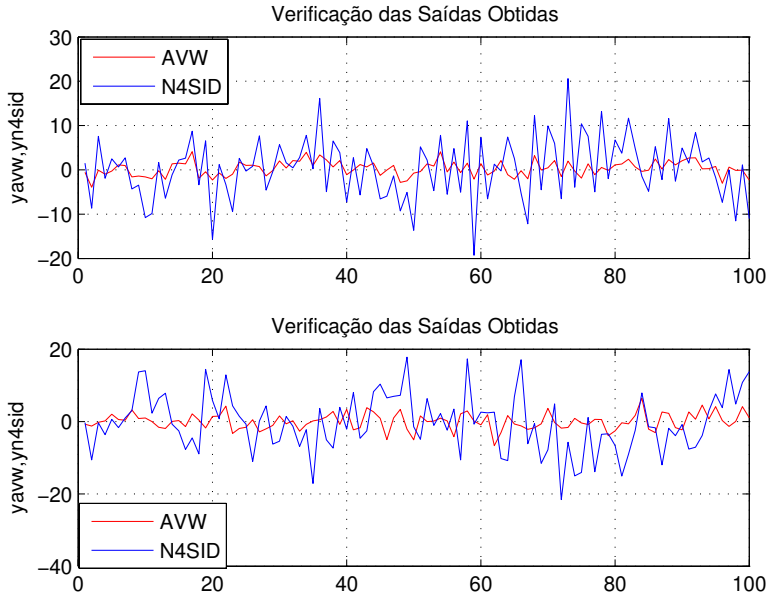

Fig. 3.17: Comparação dos Programas Moesp_Aoki e N4SID

Após a verificação dos execelentes resultados obtidos com nossa proposta de algoritmo **Moesp_Aoki**, decidimos estudar o desempenho do algoritmo **N4SID** quando aplicado de forma

similar à que propusemos para o algoritmo **AVW**, e o fazemos a seguir.

- Neste último exemplo apresentamos os resultados obtidos na identificação com o algoritmo **N4SID**, aplicado de forma similar à que fizemos na proposta **AVW**, eles foram:

$$A_c^d = \begin{bmatrix} -0.1944 & -0.9030 & 0.3592 & 0.2562 \\ 0.7961 & 0.0355 & 0.4118 & 0.4625 \\ 0.1349 & 0.3030 & -0.4674 & 0.6381 \\ -0.1794 & 0.1155 & 0.0256 & 0.3855 \end{bmatrix}, \quad B_c^d = \begin{bmatrix} -0.014366 & -0.010712 & -0.043325 \\ 0.030466 & -0.010248 & -0.013684 \\ -0.028931 & -0.027114 & 0.032767 \\ -0.015327 & -0.019649 & -0.017892 \end{bmatrix},$$

$$C_c^d = \begin{bmatrix} 3.4506 & -1.7569 & -4.8586 & -1.4837 \\ -5.3055 & 0.70354 & -3.0513 & 2.1877 \end{bmatrix}, \quad D_c^d = \begin{bmatrix} -0.26937 & 0.010749 & -1.1714 \\ -1.833 & 0.27374 & 1.1635 \end{bmatrix}$$

A Fig.3.18 apresenta a saída do modelo determinístico obtido executando-se o algoritmo **N4SID**.

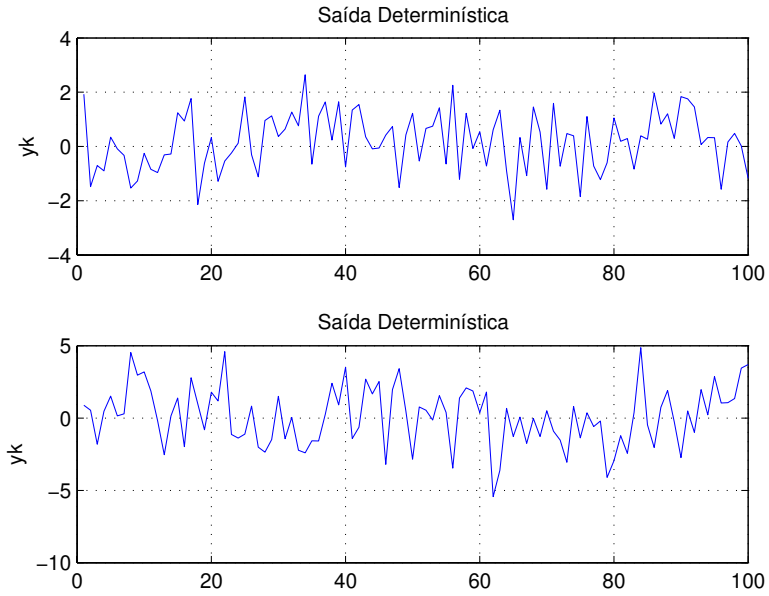

Fig. 3.18: Saída do Modelo Determinístico

A seguir determinamos um sinal que representa um ruído colorido,e executamos novamente o algoritmo **N4SID**, agora para o caso estocástico e obtemos o seguinte modelo:

$$A_c^e = \begin{bmatrix} -0.21105 & -0.98065 & 0.04824 & 0.24854 \\ 0.72368 & -0.2091 & 0.41128 & 0.28688 \\ 0.37559 & 0.35165 & -0.39035 & 0.73337 \\ -0.079532 & 0.14694 & 0.085909 & 0.43784 \end{bmatrix}$$

$$K = \begin{bmatrix} -0.022249 & -0.0019839 \\ -0.0078512 & -0.020635 \\ -0.0063208 & -0.013172 \\ -0.010384 & 0.010395 \end{bmatrix} \quad C_c^e = \begin{bmatrix} 3.2422 & -0.18929 & -4.8943 & -2.0766 \\ -5.3072 & 1.4983 & -2.8022 & 1.4242 \end{bmatrix},$$

Na Fig.3.19 apresentamos o sinal modelado usando o algoritmo **N4SID** na forma que consideramos adequada. A seguir apresentamos uma superposição dos sinais y_{kc}^{d} e y_{kc}^{e}. Finalmente fazemos uma verificação para validar do algoritmo **N4SID** e observamos na Fig.3.21 que para um conjunto de dados entrada-saída com ruído branco o algoritmo consegue descrever o sinal ruidoso y_{kr}.

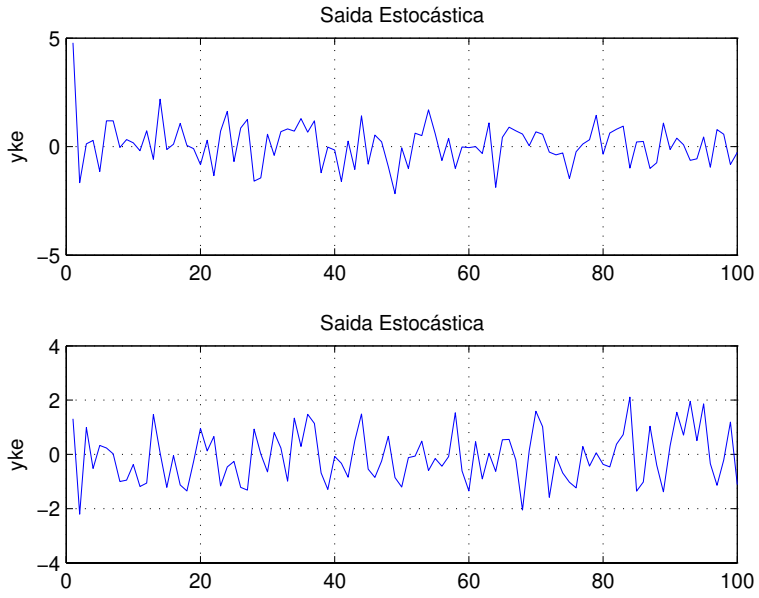

Fig. 3.19: Sinal de saída modelado

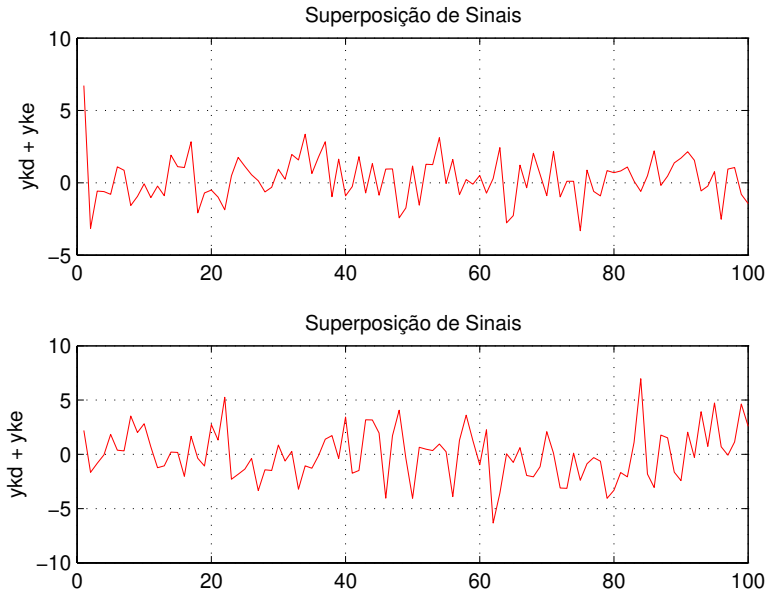

Fig. 3.20: Superposição dos Sinais obtidos pelo *N4SID*

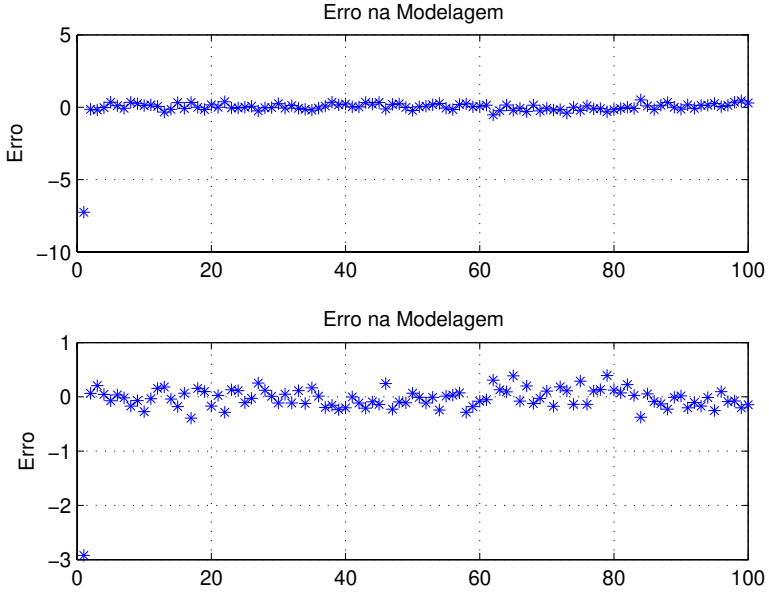

Fig. 3.21: Nova Análise de Erros do Programa *N4SID*

Capítulo 4

Soluções Neurais de Equações Algébricas de Riccati

Dentre as equações matriciais não-lineares mais estudadas e utilizadas por matemáticos e engenheiros estão as Equações de Riccati. O termo genérico **"Equação de Riccati"** pode significar qualquer classe de matrizes: quadrática, algébrica ou diferencial ou a diferenças finitas do tipo simétrico ou não-simétrico, surgida no estudo de sistemas dinâmicos contínuos ou discretos no tempo [6]. As equações de Riccati aparecem naturalmente numa ampla variedade de situações e têm grande utilidade na análise e projeto de sistemas de controle.

Estas equações (discretas e contínuas) desempenham um papel fundamental na solução de problemas de Controle Linear Quadrático Gaussiano, estimação de estado e de parâmetros de sistemas, modelagem de séries temporais multivariáveis e em muitos outros ramos da matemática aplicada. Por outro lado, muitas pesquisas têm sido reportadas para resolver sistemas de equações lineares e problemas relacionados com Redes Neurais Artificiais (RNA), [87, 169, 170], embora, exista pouca literatura tratando da solução neural da equação matricial de Riccati discreta no tempo.

Uma Rede Neural Artificial é uma estrutura de processamento de informação distribuída paralelamente na forma de um grafo direcionado, com algumas restrições e definições próprias, consistindo de neurônios com interconexões sinápticas e enlaces de ativação, vide [57].

Na literatura de controle, dá-se muita atenção aos problemas relacionados com o projeto do regulador linear quadrático (LQR). O problema de projetar um sistema de controle linear realimentado, minimizando um índice de desempenho quadrático, pode, por exemplo ser reduzido ao problema de obter uma solução definida não negativa da equação algébrica de Riccati. Apesar de existirem algoritmos paralelos que calculam a solução de Riccati mais rapidamente que os algoritmos seqüênciais, e de existirem muitas referências à pesquisas para resolver sistemas de equações lineares e problemas relacionados com *RNA*, [87, 169, 170], referências tratando da solução neural da equação matricial de Riccati discreta são escassas.

Um objetivo deste trabalho é resolver problemas de controle ótimo *LQR* para sistemas lineares discretos e contínuos no tempo utilizando *RNR*. Temos em mente, subjacente, o objetivo de criar condições para o controle em tempo real de sistemas com a intenção de que as soluções neurais já desenvolvidas e aqui implementadas por software venham a ser implementadas em hardware microeletrônico em futuro próximo.

Neste capítulo resolvemos usando *RNA* as equações algébricas de Riccati discretas (*EARD*) e

contínuas (*EARC*) no tempo. Para o caso discreto, apresentamos uma proposta de abordagem utilizando uma Rede Neural Recorrente multicamada *RNR*, [149], para resolver computacionalmente a equação algébrica de Riccati discreta no tempo (*EARD*), ou seja, obter uma única solução definida não-negativa da *EARD* usando *RNR* e comparar os resultados com os provenientes de outros métodos existentes. Propomos duas equações diferenciais não-lineares matriciais acopladas, que descrevem a dinâmica neural da equação Neuro-Riccati proposta. Estas equações matriciais acopladas são resolvidas pela RNR e pretendemos que a abordagem proposta seja capaz de obter uma solução simétrica, definida não-negativa $P \in \Re^{nxn}$ da *EARD*.

Para o caso contínuo, apoiados nos modelos dinâmicos neurais que descrevem a *EARC*, [171], fazemos uma implementação computacional que além de calibrar nossa solução neural para este caso, permite validar a proposta de abordagem discreta que fizemos e também implementamos. Apresentamos vários exemplos de aplicação ao problema de projeto de regulador linear quadrático.

4.1 Equação de Riccati Discreta no Tempo

Para motivação, consideremos o seguinte sistema controlável linear invariante e discreto no tempo, definido pela equação

$$x_{k+1} = Ax_k + Bu_k, \tag{4.1}$$

onde $x_k \in \Re^n$ é o vetor de estado do sistema e $u_k \in \Re^m$ é o vetor de entrada de controle, $A \in \Re^{n\times n}$ e $B \in \Re^{n\times m}$ são matrizes constantes conhecidas de dimensões apropriadas associadas com x_k e u_k respetivamente. A função de custo quadrática associada com este sistema pode ser definida como:

$$J = \sum_{k=0}^{\infty}(x_k^T Q x_k + u_k^T R u_k), \tag{4.2}$$

onde $Q > 0$ e $R > 0$ são matrizes de ponderação conhecidas, simétricas e definidas positivas para x_k e u_k respetivamente; esta função deve ser minimizada com a restrição (4.1). Um controlador ou lei de controle com realimentação de estado linear, $u_k = -Kx_k$ resultante, [173], pode ser aplicado na equação (4.1) e obtemos para o sistema em malha fechada a seguinte expressão:

$$x_{k+1} = (A - BK)x_k, \qquad x_0 \tag{4.3}$$

onde a matriz de ganho realimentada ótima pode ser calculada como:

$$K = (B^T P B + R)^{-1} B^T P A \tag{4.4}$$

e P é uma matriz simétrica, definida não-negativa que pode ser obtida através da solução da *EARD*:

$$A^T P(I + SP)^{-1} A - P + Q = 0 \quad , \tag{4.5}$$

onde A, Q e S são matrizes reais quadradas com Q e S simétricas e A não-singular, I corresponde à matriz identidade. A matriz S, para a equação (4.5), é definida como:

$$S \ = BR^{-1}B^T \tag{4.6}$$

e R é uma matriz não-singular,[143, 160].

4.2 Equação Dinâmica Neural de Riccati Discreta

Nesta seção, fazemos uma proposta para resolver a *EARD* definida em (4.5), usando RNR.

Apresentamos, nesta seção e na seção 5.5, análises das equações de Riccati Discreta e Contínua, respectivamente. Nosso objetivo com a análise feita com a equação de Riccati Contínua, é principalmente calibrar nossa proposta computacional, além de testar nossa lógica de raciocínio para desenvolver as equações dinâmicas neurais discretas, pois sobre a resolução desta equação contínua, existe bastante bibliografia e trabalhos desenvolvidos, em particular os relacionados com a solução neural contínua, vide [171].

A seguir fazemos alguns comentários relacionados com as definições dos problemas neurais em relação à função objetivo que deve ser minimizada. Utilizaremos uma função de energia $e(F)$ para qualquer função de ativação F. Neste nosso problema, ela será definida como: $e(F) = \frac{1}{2}F^2$.

Os resultados dos estudos realizados em [169, 170] indicam que qualquer função de otimização não decrescente em F pode ser usada. Exemplos típicos de funções de ativação (f_{ij}) e suas correspondentes funções de energia (e_{ij}) são apresentados na expressão 4.7. A relação entre a função de ativação F e a função convexa e_{ij} é dada por $\frac{\partial e_{ij}(\epsilon)}{\partial \epsilon} = f_{ij}(\epsilon)$.

$$\begin{array}{llll} f_{ij}(\xi) & = \xi & e_{ij}(\xi) & = \frac{\xi^2}{2} \\ f_{ij}(\xi) & = arctan(\xi) & e_{ij}(\xi) & = \xi arctan(\xi) - ln\sqrt{1-\xi^2} \\ f_{ij}(\xi) & = tanh(\xi) & e_{ij}(\xi) & = \xi + ln(1+exp(-2\xi)) \end{array} \tag{4.7}$$

Em vista do fato de que a *EARD* possui múltiplas soluções, nosso problema consiste na utilização da equação (4.5), e de mais uma equação que corresponde à uma restrição, para garantir a obtenção de uma solução P única, simétrica e definida positiva. Dado que qualquer matriz real, definida positiva e simétrica tem um fator Cholesky, a restrição que vamos incluir em nosso problema é definida por:

$$LL^T = P \tag{4.8}$$

onde $L \in \Re^{n \times n}$ é uma matriz triangular superior com os elementos da diagonal todos positivos.

Nosso problema neural é definido com a seguinte função critério:

$$\begin{array}{rcl} \min E(P,L) & = & \sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}[G(P)] + e_{ij}[H(P,L)] \\ & onde & \\ G(P) & = & A^T P(I+SP)^{-1}A - P + Q = 0 \\ H(P,L) & = & LL^T - P = 0 \end{array} \tag{4.9}$$

Nosso objetivo, com estas duas equações inclusas na função critério, é obter as equações dinâmicas neurais para resolver, usando RNR, o problema de controle ótimo apresentado.

As equações dinâmicas neurais para resolver este tipo de problemas em forma geral estão descritas em [87]:

$$\frac{dP(t)}{dt} = -\eta_p \frac{\partial E}{\partial P} = -\eta_p W_1 \qquad P(0) = P^T(0) \tag{4.10}$$

$$\frac{dL(t)}{dt} = -\eta_l \frac{\partial E}{\partial L} = -\eta_l W_2 \qquad W_2(0) \neq 0 \tag{4.11}$$

onde a derivada de uma função de valor escalar E em relação a uma matriz é definida por

$$\frac{\partial E}{\partial P} = \Big[\frac{\partial E}{\partial p_{ij}}\Big]_{n\times n} \qquad i,j = 1,...,n \tag{4.12}$$

Na definição das equações em (4.10) e (4.11), $P(t)$ e $L(t)$ são matrizes de ativação de estado da *RNR*, $\eta_p, \eta_l > 0$ são as taxas de aprendizado e $W_1 = [w_{1,ij}]_{n\times n}$ e $W_2 = [w_{2,ij}]_{n\times n}$ são definidas para todo $i,j = \overline{1,n}$ em [149].

Antes de continuar com a apresentação da solução numérica do problema neural exposto, vamos apresentar algumas operações com derivadas úteis na solução de expressões definidas em relação a uma matriz solução P qualquer. Isto é necessário, pois temos que trabalhar com derivadas parciais de matrizes em relação a outra matriz solução. A seguir apresentamos o seguinte Lema.

Lema 1 *Suponha dada uma função de energia $E = E[G(X)]$, uma matriz solução $X \in \Re^{n\times n}$ e uma matriz de ativação não-decrescente $F = [f_{ij}(g_{ij})]$. As seguintes relações são satisfeitas pelas derivadas:*

1. $G(X) = A^T X, A \in \Re^{l\times n}$ *então* $\frac{\partial E}{\partial X} = F^T A^T, F \in \Re^{l\times n}$
2. $G(X) = XA, A \in \Re^{n\times l}$ *então* $\frac{\partial E}{\partial X} = FA^T, F \in \Re^{m\times l}$
3. $G(X) = X^T A, A \in \Re^{m\times l}$ *então* $\frac{\partial E}{\partial X} = AF^T, F \in \Re^{n\times l}$
4. $G(X) = AXH^T, A \in \Re^{n\times n}, H \in \Re^{n\times n}$ *então* $\frac{\partial E}{\partial X} = A^T FH, F \in \Re^{n\times n}$
5. $G(X) = A^{-1}X, A \in \Re^{n\times n}$ *então* $\frac{\partial E}{\partial X} = -A^{-1}FA^{-1}, F \in \Re^{n\times n}$

Prova 1 *A prova deste lema pode ser vista em [87].*

Com isto teremos ferramentas necessárias para poder continuar nosso desenvolvimento para obtenção das respectivas equações dinâmicas neurais.

Voltando ao nosso problema usando RN, vamos redefinir as equações em (4.9) para $G(P)$ e $H(P,L)$ em forma de somatórias e escolher $P = [v_{ij}(t)]$ e $L = [z_{ij}(t)]$, $\forall\ i,j = \overline{1,n}$, tal que

$$\begin{aligned} g_{ij}(v_{ij}(t)) &= \sum_{k=1}^{n}\sum_{l=1}^{n} a_{ki}v_{kl}(t)(I_{ij} + s_{ik}v_{kj}(t))^{-1}a_{lj} - v_{ij}(t) + q_{ij} \\ h_{ij}(v_{ij}(t), z_{ij}(t)) &= \sum_{k=1}^{n} z_{ki}(t)z_{kj}(t) - v_{ij}(t) \end{aligned} \tag{4.13}$$

Com base nas equações definidas em (4.10) e (4.11), e no Lema 1, podemos obter o conjunto de equações dinâmicas neurais que vão resolver a equação (4.9) como:

$$\begin{aligned} \tfrac{dv_{ij}(t)}{dt} &= -\eta_p\Big[\sum_{k=1}^{n}\sum_{l=1}^{n}\frac{\partial e_{1,kl}(g_{kl})}{\partial g_{kl}}\frac{\partial g_{kl}}{\partial v_{ij}} + \sum_{k=1}^{n}\sum_{l=1}^{n}\frac{\partial e_{2,kl}(h_{klj})}{\partial h_{kl}}\frac{\partial h_{kl}}{\partial v_{ij}}\Big] \\ \tfrac{dz_{ij}(t)}{dt} &= -\eta_l\Big[\sum_{k=1}^{n}\sum_{l=1}^{n}\frac{\partial e_{2,kl}(h_{kl})}{\partial h_{kl}}\frac{\partial h_{kl}}{\partial v_{ij}}\Big] \end{aligned} \tag{4.14}$$

A relação entre a função de ativação não-decrescente F e a função de energia convexa e_{ij} pode ser escrita como, [171]

$$\begin{array}{ll} \frac{\partial e_{1,kl}(g_{kl})}{\partial g_{kl}} & = f_{1,kl}(g_{kl}) \\ \frac{\partial e_{2,kl}(h_{kl})}{\partial h_{kl}} & = f_{2,kl}(h_{kl}) \end{array} \tag{4.15}$$

Pela própria definição da função de energia $e_{s,kl}(g_{kl})$, podemos adotar que $f_{1,kl}(g_{kl}) = g_{kl}$ e $f_{2,kl}(h_{kl}) = h_{kl}$. Logo as equações (4.14) e (4.14) ficam:

$$\begin{array}{ll} \frac{dv_{ij}(t)}{dt} & = -\eta_p \Big[\sum_{k=1}^{n} \sum_{l=1}^{n} f_{1,kl}(g_{kl}) \frac{\partial g_{kl}}{\partial v_{ij}} + \sum_{k=1}^{n} \sum_{l=1}^{n} f_{2,kl}(h_{kl}) \frac{\partial h_{kl}}{\partial v_{ij}} \Big] \\ \frac{dz_{ij}(t)}{dt} & = -\eta_l \Big[\sum_{k=1}^{n} \sum_{l=1}^{n} f_{2,kl}(h_{kl}) \frac{\partial h_{kl}}{\partial z_{ij}} \Big] \end{array} \tag{4.16}$$

onde as taxas de aprendizado (η_p, η_l) são positivas e vão definir a velocidade da convergência na solução; logo, a convergência do processo de computação neural pode ser acelerada selecionando o par (η_p, η_l) com valores suficientemente grandes e tal que se, por exemplo quisermos garantir que $P(t)$ seja definida positiva, será necessário que $L(t)$ converja mais rápido do que $P(t)$ e para isto escolheremos $(\eta_p \leq \eta_l)$. Dado que a faixa de convergência da *RNR* é incrementada em função dos incrementos das constantes η_p e η_l, a convergência da *RNR* pode ser acelerada pela escolha de constantes η_p e η_l suficientemente grandes.

As matrizes de ativação são definidas como:

$$\begin{array}{lll} f_{1,kl}(g_{kl}) & = U(t) = & F(A^T P(I + SP)^{-1} A - P + Q) \\ f_{2,kl}(h_{kl}) & = Y(t) = & F(LL^T - P) \end{array}$$

Vamos começar os cálculos dos termos nas equações (4.16) e (4.16) em forma matricial, para poder chegar às equações dinâmicas neurais, que é o objetivo principal deste capítulo.

$$\begin{array}{ll} \sum_{k=1}^{n} \sum_{l=1}^{n} \frac{\partial g_{kl}}{\partial v_{ij}} & = \frac{\partial (A^T V (I+SV)^{-1} A - V + Q)}{\partial V} \\ \sum_{k=1}^{n} \sum_{l=1}^{n} \frac{\partial h_{kl}}{\partial v_{ij}} & = \frac{\partial (ZZ^T - V)}{\partial V} \\ \sum_{k=1}^{n} \sum_{l=1}^{n} \frac{\partial h_{kl}}{\partial z_{ij}} & = \frac{\partial (ZZ^T - V)}{\partial Z} \qquad \forall \ i, j = 1, \ldots, n \end{array} \tag{4.17}$$

Trabalhando com a equação (4.17) e aplicando as regras para derivada da soma e para derivada do produto de matrizes, definidas no Lema 1, podemos escrever:

$$\begin{array}{ll} \frac{\partial G}{\partial V} & = \frac{\partial A^T V}{\partial V} (I + SV)^{-1} A + A^T V \frac{\partial (I+SV)^{-1}}{\partial V} A - \frac{\partial V}{\partial V} \\ \frac{\partial (I+SV)^{-1}}{\partial V} & = -(I + SV)^{-1} \frac{\partial (I+SV)}{\partial V} (I + SV)^{-1} \end{array} \tag{4.18}$$

Desta forma obtivemos as seguintes equações dinâmicas neurais para resolver a *EARD*, (4.5), uti-

lizando uma *RNR*

$$\begin{array}{rcl} \frac{dV(t)}{dt} = & -\eta_v[U(t)^T A^T (I+SV(t))^{-1}A - A^T V(t)(I+SV(t))^{-1}S^T U(t) \\ & (I+SV(t))^{-1}A - U(t) - Y(t)] \\ \frac{dZ(t)}{dt} = & -\eta_z Z(t)Y^T(t) \\ U(t) = & F(A^T V(t)(I+SV(t))^{-1}A - V(t) + Q) \\ Y(t) = & F(Z(t)Z^T(t) - V(t)) \end{array} \tag{4.19}$$

Os resultados da implementação computacional utilizando uma *RNR* são apresentados na seção 4.5.

4.3 Equação Dinâmica Neural de Riccati Contínua

Nesta seção apresentamos alguns detalhes para a obtenção das equações dinâmicas neurais que resolvem a equação de Riccati contínua utilizando *RNR*, que ajudam na implementação computacional e na comparação dos resultados com métodos similares apresentados em alguns artigos estudados, relacionados com a solução neural da equação de Riccati contínua bem como na calibração da implementação proposta para *EARC* neural. Para isto nos apoiaremos na referência [171], que foi muito importante no desenvolvimento desta parte da tese.

Considere um sistema linear estacionário contínuo de estado completamente controlável:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{4.20}$$

cuja lei de controle, $u(t) = -Kx(t)$, depende de uma matriz de ganho, K, que vai estar definida através de uma matriz P correspondente à solução da equação matricial algébrica de Riccati Contínua:

$$A^T P + PA - PSP + Q = 0 \tag{4.21}$$

onde $S = BR^{-1}B^T$ é uma matriz simétrica e semidefinida positiva.

A função de custo quadrática associada com este sistema pode ser definida como:

$$J = \int_{k=0}^{\infty} (x(t)^T Q x(t) + u(t)^T R u(t))dt, \tag{4.22}$$

onde $Q > 0$ e $R > 0$ são matrizes de ponderação conhecidas, simêtricas e definidas positivas para $x(t)$ e $u(t)$ respetivamente; esta função deve ser minimizada com a restrição (5.27).

Para fazer a formulação utilizando uma *RNR* na solução da equação (4.21), retomamos apenas as equações que foram definidas para o caso discreto, no seção anterior.

Seja a função objetivo definida por:

$$\begin{array}{rcl} \min E(P,L) & = & \sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}[G(P)] + e_{ij}[H(P,L)] \\ & \text{onde} & \\ G(P) & = & PSP - A^T P - PA - Q \\ H(P,L) & = & LL^T - P \end{array} \tag{4.23}$$

cujas equações em forma de somatória são definidas a seguir, onde adotamos as notações $P = v(t)$ e $L = z(t)$:

$$\begin{aligned} g_{ij}(v_{ij}(t)) &= \sum_{k=1}^{n}\sum_{\substack{l=1 \\ min\{i,j\}}}^{n} v_{ik}(t)s_{kl}v_{lj}(t) - \sum_{k=1}^{n}[a_{ki}v_{ik}(t) + v_{ik}(t)a_{kj}] - q_{ij} \\ h_{ij}(v_{ij}(t), z_{ij}(t)) &= \sum_{k=1} [z_{ik}(t)z_{kj}(t) - v_{ij}(t)] \end{aligned} \tag{4.24}$$

As equações dinâmicas neurais respectivas são:

$$\begin{aligned} \frac{dv_{ij}(t)}{dt} &= -\eta_p \frac{\partial E(v,z)}{\partial v_{ij}} = -\eta_p \left[\sum_{k=1}^{n}\sum_{l=1}^{n} \frac{\partial g_{kl}}{\partial v_{ij}} u_{kl}(t) + \frac{\partial h_{kl}}{\partial v_{ij}} y_{kl}(t) \right] \\ \frac{dz_{ij}(t)}{dt} &= -\eta_l \frac{\partial E(v,z)}{\partial z_{ij}} = -\eta_l \left[\sum_{k=1}^{n}\sum_{l=1}^{n} \frac{\partial h_{kl}}{\partial z_{ij}} y_{kl}(t) \right] \end{aligned} \tag{4.25}$$

As funções de ativação são definidas como:

$$\begin{aligned} f_{1,kl}(g_{kl}) &= U(t) = F(V(t)SV(t) - A^T V(t) - V(t)A - Q) \\ f_{2,kl}(h_{kl}) &= Y(t) = F(Z(t)Z^T(t) - V(t)) \end{aligned}$$

Desta forma, obtemos as seguintes equações dinâmicas neurais que resolvem a equação (4.21) utilizando uma *RNR*:

$$\begin{aligned} \frac{dV(t)}{dt} &= -\eta_p[V(t)SU(t) + U(t)SV(t) - AU(t) - U(t)A^T - Y(t)] \\ \frac{dZ(t)}{dt} &= -\eta_l Y(t)Z(t) \\ U(t) &= F(V(t)SV(t) - A^T V(t) - V(t)A - Q) \\ Y(t) &= F(Z(t)Z^T(t) - V(t)) \end{aligned} \tag{4.26}$$

4.4 Estrutura da Rede Neural

As arquiteturas das nossas implementações *RNR*, são constituídas de quatro camadas conectadas bidirecionalmente. Para o caso discreto, por exemplo, a camada 1 (ou primeira) é de entrada, representada por $U(t) = [u_{ij}(t)]$, a camada 4 (ou última) de saída, representada por $V(t) = [v_{ij}(t)]$ e duas camadas intermédiarias representadas por $Y(t) = [y_{ij}(t)]$ e $Z(t) = [z_{ij}(t)]$, respectivamente. A partir disso, podemos fazer P corresponder à saída final da rede ($v_{ij}(t)$). A matriz de ativação de estado $V(t)$ para os neurônios na camada de saída representa o resultado computacional de P (solução da equação (4.5)) e a matriz de estado $Z(t)$ representa o fator de Cholesky de P, ou seja L. As camadas para $V(t), U(t)$ e $Y(t)$ vão consistir de arranjos quadrados $(n \times n)$ de neurônios.

4.5 Implementação e Resultados

Nesta seção, discutimos os resultados das implementações usando Redes Neurais das equações algébricas de Riccati discreta (*EARD*) e contínua (*EARC*) com vários exemplos. As implementações para resolver a **EARD Neural** e a **EARC Neural** em forma geral, foram desenvolvidas no Matlab

usando o método de Runge-Kutta de quarta ordem; em nosso caso as equações diferenciais coincidem com as equações dinâmicas neurais obtidas em cada problema analisado: Discreto e Contínuo, respectivamente. A estrutura do algoritmo para calcular $V(t)$, por exemplo, no caso discreto é:

s_1 = feval('fv',etav,V,S,U,A);

s_2 = feval('fv',etav,V+h*s_1/2,S,U,A);

s_3 = feval('fv',etav,V+h*s_2/2,S,U,A);

s_4 = feval('fv',etav,V+h*s_3,S,U,A);

V = V + h*(s_1 + 2*s_2 + 2*s_3 + s_4)/6;

Cada s_i corresponde a uma variável que é encarregada da atualização de um peso sináptico da rede ou equivalentemente à variável desconhecida do problema, x.

Exemplo 1. Considere o controle LQR do seguinte sistema contínuo no tempo, instável em malha aberta, onde os coeficientes das matrizes são:

$$A = \begin{bmatrix} 0 & 1 \\ -2 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1.6 & 0.9 \\ -0.1 & 2.1 \end{bmatrix} \quad Q = \begin{bmatrix} 1.5 & -1 \\ -1 & 5 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.27}$$

Resolvemos a equação neural de Riccati Contínua com a mesma arquitetura da *RNR* e com a implementação do algoritmo numérico da *RNR* explicado detalhadamente em [171]. A solução da respectiva equação é:

$$P = \begin{bmatrix} 1.222 & -0.671 \\ -0.671 & 1.339 \end{bmatrix}$$

A simulação foi realizada usando os seguintes parâmetros como condição inicial, $f_{kl}(\xi) = \xi$,

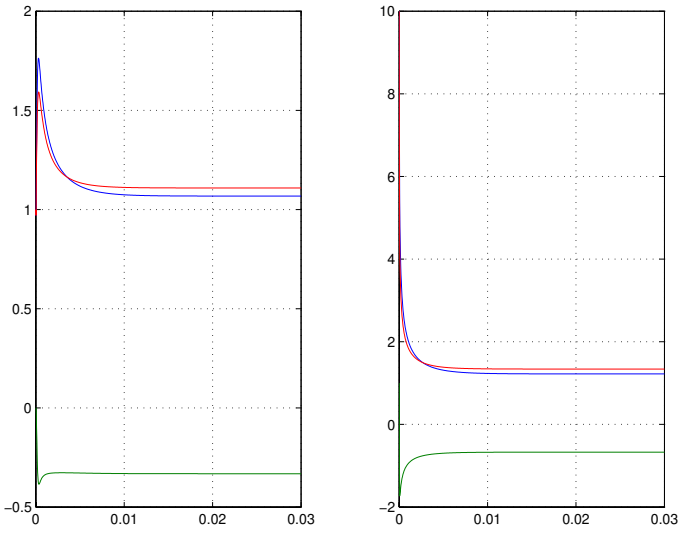

Fig. 4.1: Resultados da equação neural de Riccati Contínua

$(\eta_p, \eta_l) = (10, 1000)$, $\Delta t = 10^{-5}$, $P(0) = \begin{bmatrix} 10 & 1 \\ 1 & 10 \end{bmatrix}$ and $Y(0) = I$.

A Figura 4.1 apresenta as trajetórias de $Z(t)$ e $V(t)$, respectivamente. A partir destas figuras podemos observar que tanto $Z(t)$ como $V(t)$ atingem seus valores permanentes rapidamente.

Neste exemplo estamos simplesmente verificando e calibrando nossa implementação computacional da *EARC* neural para futuras utilizações em outras equações, por exemplo as equações de Lyapunov. Temos como referência o artigo [171], onde a simulação, para o mesmo exemplo, foi feita em código C. Neste caso fizemos uma comparação entre os resultados obtidos no trabalho de referência e os obtidos em nossa implementação usando RNR e chegamos a conclusão que a nossa proposta utilisando redes Neurais também está correta.

Exemplo 2. Este exemplo considera o problema de controle ótimo linear quadrático para o caso discreto. O desempenho da solução da **EARD**, usando uma abordagem com rede neural recorrente multicamada é apresentado. Os parâmetros para as matrizes do sistema são:

$$A = \begin{bmatrix} 0.9512 & 0 \\ 0 & 0.9048 \end{bmatrix} \quad B = \begin{bmatrix} 4.877 & 4.877 \\ -1.1895 & 3.569 \end{bmatrix} \quad Q = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.02 \end{bmatrix} \quad R = \begin{bmatrix} 0.33 & 0 \\ 0 & 3 \end{bmatrix}$$

As equações dinâmicas neurais (4.19) foram utilizadas para resolver a equação de Riccati Discreta (4.5). A solução P, K obtida para a respectiva **Neural-DARE** é:

$$P = \begin{bmatrix} 0.0104 & 0.0032 \\ 0.0032 & 0.0504 \end{bmatrix} \qquad K = \begin{bmatrix} 0.0715 & -0.0705 \\ 0.0136 & 0.0455 \end{bmatrix}$$

Esta matriz P é simétrica, definida positiva e satisfaz (4.5). A simulação foi realizada usando as equações dinâmicas neurais implementadas, onde $f_{kl}(\xi) = \xi$, $(\eta_p, \eta_l) = (3000, 10000)$, $\Delta t = 10^{-5}$ $P(0) = [0.1]_{2\times 2}$ e $Y(0) = I$. As Figs.4.2-4.4 apresentam as trajetórias de $Z(t), V(t), K(t), x(t), u(t)$ para a solução da equação **EARD Neural** implementada neste exemplo . A partir da Fig. 4.2, podemos observar que os elementos de $V(t)$ atingem seus valores permanentes rapidamente. Esta solução utilizando *RNR* corresponde à solução dada por Laub em [79, 76], cujo método é uma variante da abordagem de autovetor clássica usando um conjunto de vetores Schur.

Portanto, nossa proposta e implementação para resolver a **EARD** utilizando *RNR*, apresenta bons resultados que podem ser testados e comprovados com qualquer outro método de solução, em especial [143, 22].

Exemplo 3. Considere o problema de controle LQR de um sistema com parâmetros variantes no tempo definidos na matrix B com taxa limitada, vale ressaltar que o elemento variante no tempo b_{11} na prática não pode ser determinado a priori. Isto dificulta determinar a lei de controle ótimo. B muda e, consequentemente, Z, V, K mudam também para satisfazer (4.5)

$$A = \begin{bmatrix} 0.9512 & 0 \\ 0 & 0.9048 \end{bmatrix} \quad Q = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.02 \end{bmatrix} \qquad B = \begin{bmatrix} 4.877 + 2sin((2\pi t)/5) & 4.877 \\ -1.1895 & 3.569 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.33 & 0 \\ 0 & 3 \end{bmatrix}; \qquad x_0 = \begin{bmatrix} 0.9516 \\ 0.2603 \end{bmatrix}$$

Sejam as matrices iniciais $V = 0$ e $Z = I$, x_0 é um vetor de estado inicial aleatório.

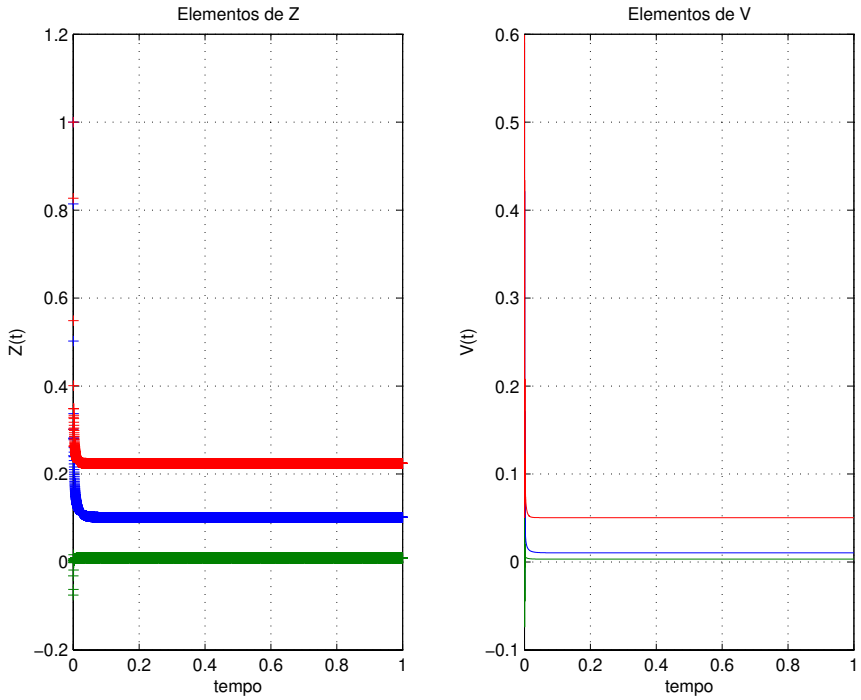

Fig. 4.2: Resultados de V(t) e Z(t), para a EARD usando *RNR*

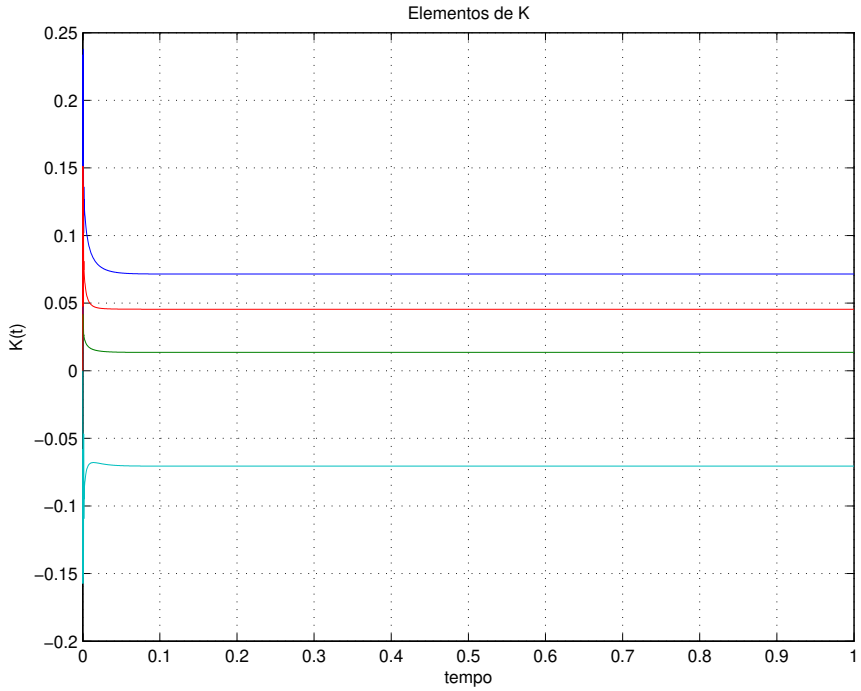

Fig. 4.3: Resultados de K(t), para a EARD usando *RNR*

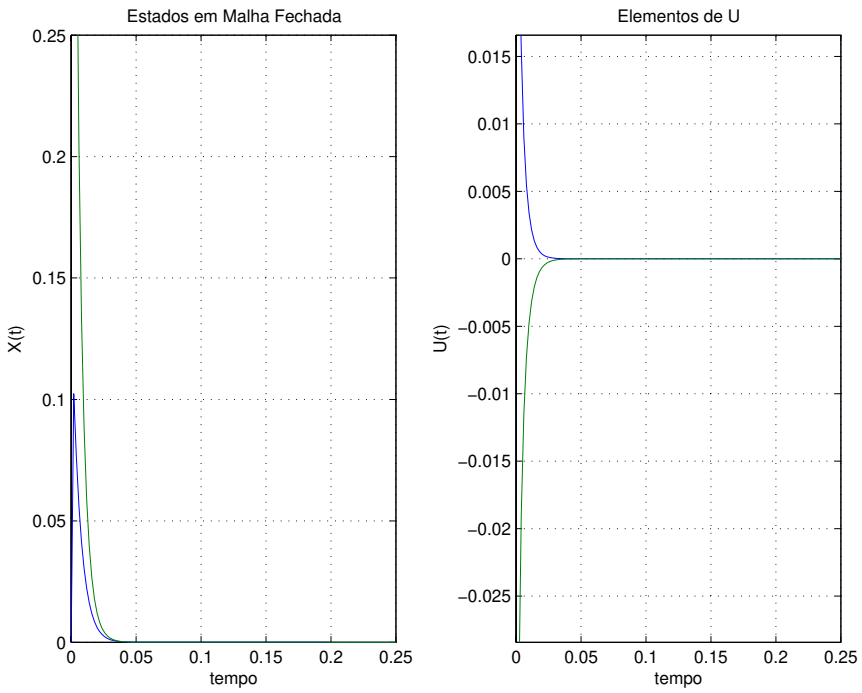

Fig. 4.4: Resultados de x(t) e u(t), usando *RNR*

A solução P, K obtida para a respectiva **EARD Neural** é:

$$P = \begin{bmatrix} 0.0098 & 0.0032 \\ 0.0032 & 0.0506 \end{bmatrix} \qquad K = \begin{bmatrix} 0.0687 & -0.0715 \\ 0.0131 & 0.0455 \end{bmatrix}$$

Simulações foram realizadas usando as equações dinâmicas neurais implementadas, com $(\eta_p, \eta_l) = (3000, 10000)$. Como $B(t)$ é uma matriz variante no tempo, resultados de simulações foram apresentados nas Figs.4.5-4.8, ilustrando os resultados de $Z(t)$, $V(t)$, $B(t)$, $K(t)$, $x(t)$ e $u(t)$. A rede neural proposta pode seguir as variações paramétricas do sistema de modo a minimizar $x(t)$ e $u(t)$.

Simulações foram realizadas usando as equações dinâmicas neurais implementadas, com $(\eta_p, \eta_l) = (3000, 10000)$. Como $B(t)$ é uma matriz variante no tempo, resultados de simulações são apresentados nas Figs.4.5-4.7, ilustrando os resultados de $Z(t)$, $V(t)$, $B(t)$, $K(t)$, $x(t)$ e $u(t)$. Pode-se observar que apesar dos valores da matrix $B(t)$ mudarem, as demais matrizes, $Z(t)$, $V(t)$ e $K(t)$ acompanham as mudanças de forma a satisfazer (4.5).

Na segunda figura em Fig.4.5 apresentamos uma comparação entre a solução obtida para a equação **EARD Neural** a cada $0.05s$ com a solução exata da equação algébrica de Riccati discreta para este caso analisado. A partir desta mesma Figura, podemos observar como os elementos de ambas matrizes $Z(t)$ e $V(t)$ atingem seus valores permanentes rapidamente.

A cada $0.05s$ temos um modelo novo da planta. Este modelo é obtido, deixando a matriz $B(t)$ fixa durante este intervalo de tempo e fazendo 10 iterações para resolver a *EARD Neural*. Logo, a cada $0.05s$ obtém-se uma nova solução *EARD Neural* em $V(t)$, calcula-se então o ganho K e aplica-se a lei de controle realimentado em malha fechada na planta. A cada $0.05s$ temos um novo estado da planta.

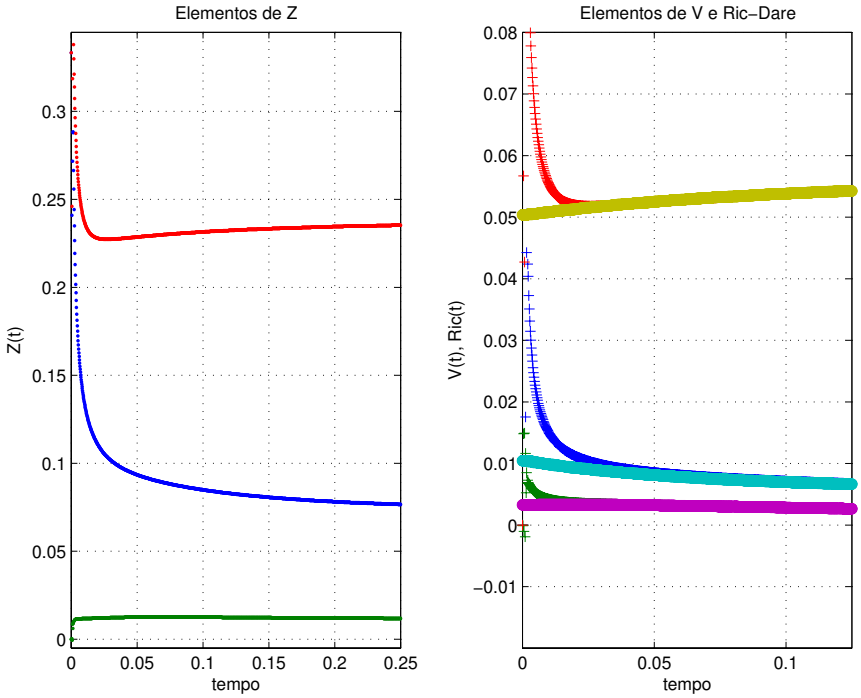

Fig. 4.5: Resultados de Z(t) e V(t) usando *RNR*

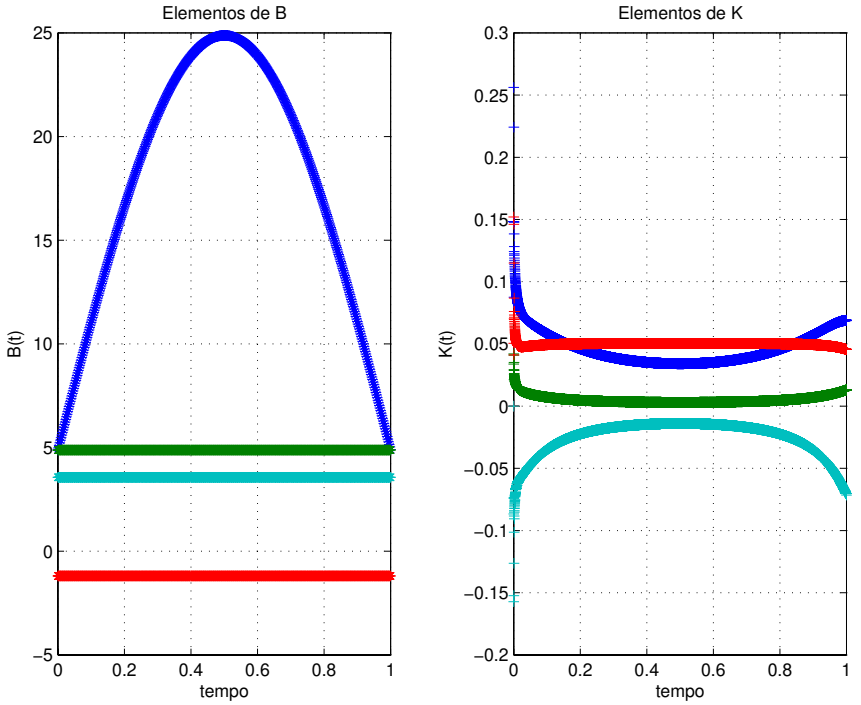

Fig. 4.6: Resultados de B(t) e K(t) usando *RNR*

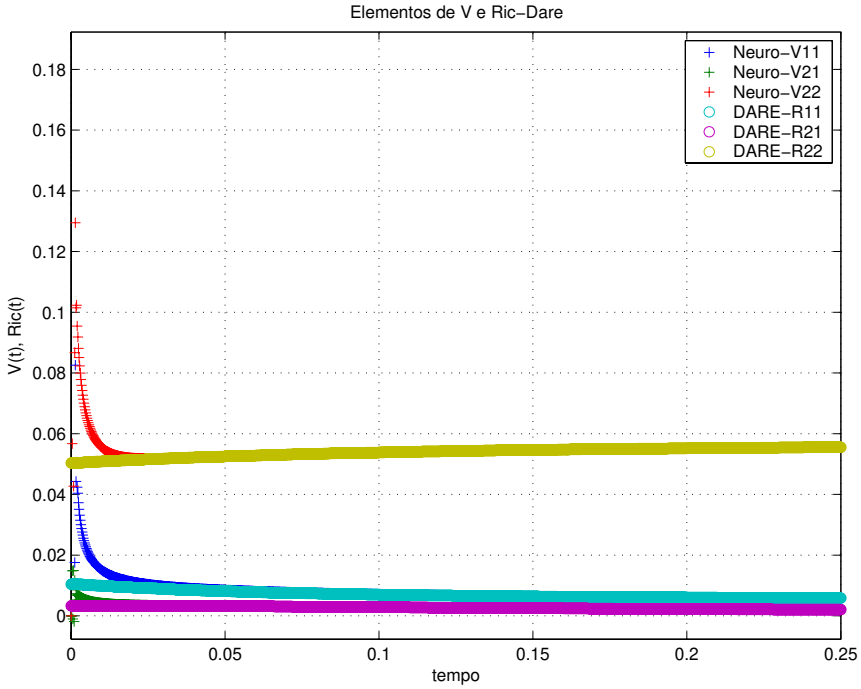

Fig. 4.7: Resultados de V(t) com *RNR*

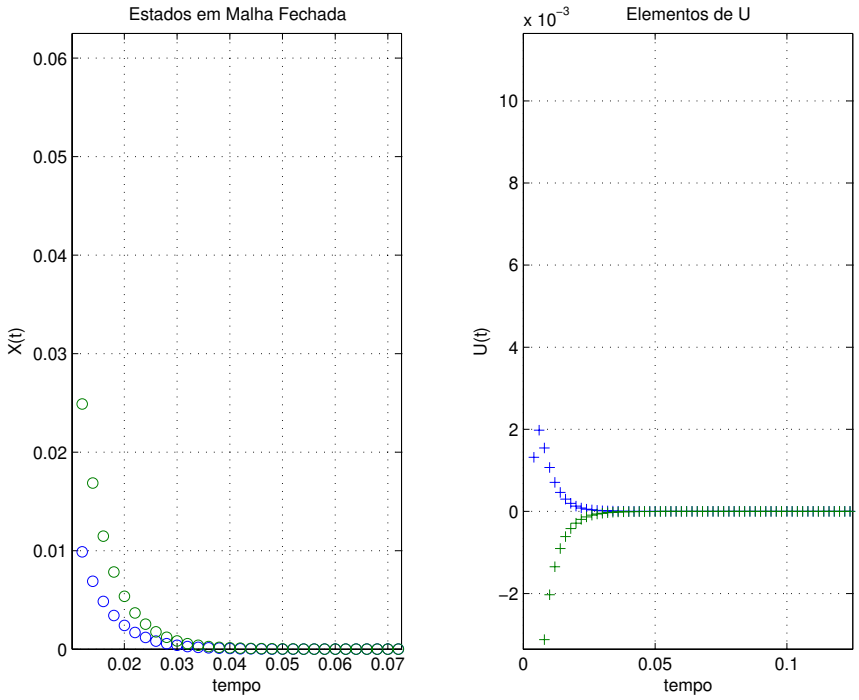

Fig. 4.8: Resultados de x(t) e u(t) usando *RNR*

A Fig.4.8 apresenta respectivamente, o vetor de estado $x(t)$ e a trajetória do vetor de controle $u(t)$ no sistema de controle realimentado. Vale ressaltar que o vetor $u(0)$ é não-nulo, pois ele é obtido a partir de uma solução da *EARD Neural* $V(t)$ iterada durante $0.05s$.

Capítulo 5

Desigualdades Matriciais Lineares: Proposta de Solução Neural da EARD

Neste Capítulo apresentamos propostas para resolver via RNA as inequações matriciais lineares (*LMI*) algébricas de Riccati discreta (*IARD*) e contínua (*IARC*) no tempo. Para o caso discreto, nos baseamos em nossa proposta de abordagem que obtém o modelo dinâmico neural que descreve a *IARD* utilizando uma *RNR*, [150]. Para o caso contínuo, apoiados em modelo dinâmico neural que descreve a *IARC*, [87], fazemos uma implementação computacional que além de calibrar nossa solução neural para este caso, permite validar a proposta de abordagem discreta que fizemos e também implementamos. Vários exemplos de aplicação em controle ótimo são apresentados.

Para cumprir nosso objetivo, é necessário introduzir alguns conceitos, fundamentações teóricas, definições que auxiliem na construção de uma proposta de solução da *IARD* que seja obtida por meio da utilização de *RNR*.

5.1 Introdução

Muitos problemas de otimização em projetos de controle e de identificação, podem ser formulados ou reformulados usando *LMI* [133]. Certamente, só faz sentido enquadrar estes problemas em termos de *LMI*, se as desigualdades puderem ser resolvidas eficientemente e de uma maneira confiável.

Na literatura de controle, dá-se muita atenção aos problemas relacionados com os projetos de regulador linear quadrático (LQR) e de controle H_∞ [171], devido sobretudo à estrutura formal destes problemas, à tratabilidade da solução e às propriedades de robustez em relação as grandes variações dos parâmetros do sistema. O problema de projetar um sistema de controle linear realimentado, minimizando um índice de desempenho quadrático, pode, por exemplo, ser reduzido ao problema de obter uma solução definida não-negativa da equação algébrica de Riccati, como vimos no Capítulo 4.

Atualmente as Inequações Matriciais Lineares (LMIs) são utilizadas como ferramentas básicas para análise e projeto de sistemas de controle de formas similares às que as equações de Lyapunov e Riccati desempenharam a partir dos anos 1960. Elas constituem uma eficiente técnica de formulação e de projeto para uma variedade de problemas de controle linear [10]. A construção de objetivos para projeto em teoria de sistemas e controle pode frequentemente ser colocada ou recolocada como problemas de programação semidefinida, ou seja, como problemas de Inequações Matriciais Lineares.

Por exemplo a análise de robustez, o projeto de controlador robusto, o projeto de controlador com escalonamento de ganho, o projeto de alocação de pólos com restrições, a análise de estabilidade de sistemas de controle fuzzy [87], etc. Apesar de existirem algoritmos para processamento paralelo que calculam a solução da Equação de Riccati mais rapidamente do que os algoritmos seqüênciais, e de existirem muitas referências a pesquisas para resolver sistemas de inequações lineares e problemas relacionados com *RNA*, [87, 170, 169], referências tratando da solução neural de inequações matriciais de Riccati são escassas, especialmente para a discreta.

Contrariamente ao objetivo de resolver numericamente problemas convexos ou quase convexos de otimização associados ao projeto de sistemas de controle envolvendo equações e inequações de Riccati ou seja, equações matriciais lineares (*EMLs*) e inequações matriciais lineares (*LMIs*) usando, por exemplo métodos de pontos interiores, nosso objetivo neste trabalho de tese é resolver tais problemas para sistemas lineares discretos e contínuos no tempo, utilizando redes neurais recorrentes. Temos em mente o objetivo de criar condições para a exploração da importância das *EMLs* e *LMIs* no controle em tempo real de sistemas tendo em conta que as soluções neurais já desenvolvidas e aqui implementadas por software sejam posteriormente implementadas em hardware microeletrônico.

Sistemas de equações diferenciais não-lineares matriciais acopladas são obtidos, descrevendo a dinâmica neural da equação *Neuro-LMI-Riccati* proposta. Estas equações matriciais acopladas são resolvidas pela RNR e pretendemos que a abordagem proposta seja capaz de obter uma solução simétrica, definida não-negativa $P \in \Re^{nxn}$ da *IARD*. Vários exemplos demonstram a efetividade da proposta e da respectiva implementação.

A utilização de *LMI* na área de análise de sistemas dinâmicos foi iniciada no final do século XIX com a publicação do trabalho de Lyapunov, que décadas após veio se tornar a base de inúmeros métodos na área de controle. Basicamente, a equação

$$\dot{x} = Ax(t) \tag{5.1}$$

descreve um sistema assintoticamente estável, se e somente se existe uma matriz simétrica P (variável livre), $P = P^T > 0$ (significando, P é uma matriz definida positiva, o que implica que $u^T Pu > 0$, $\forall u \neq 0$), tal que

$$A^T P + PA < 0, \qquad P > 0 \tag{5.2}$$

O conjunto das duas desigualdades acima é o que atualmente é chamado desigualdade de Lyapunov em P. Essa foi, historicamente, a primeira *LMI* associada ao problema de controle:

$$P - A^T P - PA > 0 \tag{5.3}$$

Nosso estudo concentra-se num sistema linear discreto e invariante no tempo descrito pelo modelo no espaço de estado:

$$\begin{cases} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{cases} \tag{5.4}$$

em que $x \in \Re^n$ representa o estado, $u \in \Re^m$ representa a entrada e $y \in \Re^l$ representa a saída.Neste modelo, as matrizes A, B, C e D possuem dimensões apropriadas.

Neste Capítulo é mostrado como implementamos e resolvemos o problema das LMIs discretas e contínuas, enquanto que no Capítulo 4 em [153, 87] o problema das EMLs foi resolvido.

Resultados básicos

A seguir, são apresentados alguns lemas e procedimentos básicos, que serão usados em diversas oportunidades no encaminhamento das soluções dos problemas apresentados neste Capítulo. Começaremos pelo lema conhecido como Complemento de Schur, muito útil para transformar certas formas não-lineares porém convexas de desigualdades em *LMI*.

Lema 2 *[108] Admita que $Q(x)$ e $R(x)$ são matrizes simétricas e que $S(x)$ depende por afinidade de x. Então a LMI*

$$\left[\begin{array}{cc} Q(x) & S(x) \\ S^T(x) & R(x) \end{array} \right] > 0 \tag{5.5}$$

é equivalente à

$$\begin{array}{c} R(x) > 0 \\ Q(x) - S(x)R^{-1}S^T(x) > 0 \end{array} \tag{5.6}$$

ou ainda à

$$\begin{array}{c} Q(x) > 0 \\ R(x) - S^T(x)Q^{-1}S(x) > 0 \end{array} \tag{5.7}$$

O Lema2 é normalmente usado para converter uma forma quadrática, como as duas últimas apresentadas acima, em uma *LMI*, conforme pode ser visto no exemplo a seguir. Vamos considerar a desigualdade de Riccati contínua

$$\begin{array}{c} R > 0 \\ Q \geq 0 \\ A^TP + PA + PBR^{-1}B^TP + Q < 0 \end{array} \tag{5.8}$$

que é quadrática em P; podemos reescrevê-la aplicando o complemento de Schur, como:

$$\left[\begin{array}{cc} -A^TP - PA - Q & PB \\ B^TP & R \end{array} \right] < 0 \tag{5.9}$$

A desigualdade de Lyapunov estabelece na verdade o conceito de estabilidade de sistemas de um modo mais geral que o conceito tradicional, uma vez que pode ser estendida para incluir sistemas não-lineares.

Em [108] podemos achar outros resultados importantes.

5.2 Equação de Riccati Discreta no Tempo

Para motivação consideremos o seguinte sistema controlável linear invariante e discreto no tempo, definido pela equação

$$x_{k+1} = Ax_k + Bu_k, \tag{5.10}$$

onde $x_k \in \Re^n$ é o vetor de estado do sistema e $u_k \in \Re^m$ é o vetor de entrada de controle, $A \in \Re^{n\times n}$ e $B \in \Re^{m\times m}$ são matrizes constantes conhecidas de dimensões apropriadas associadas com x_k e u_k respectivamente. A função de custo quadrática para este sistema é,

$$J = \sum_{k=0}^{\infty}(x_k^TQx_k + u_k^TRu_k), \tag{5.11}$$

onde $Q > 0$ e $R > 0$ são matrizes de ponderação conhecidas, simétricas e definidas positivas para x_k e u_k respectivamente; esta função deve ser minimizada com a restrição (5.10). Uma lei de controle com realimentação de estado linear, $u_k = -Kx_k$ resulta [173]. Aplicando este controlador na equação (5.10) obtemos o seguinte sistema em malha fechada:

$$x_{k+1} = (A - BK)x_k, \qquad x_0 \tag{5.12}$$

onde a matriz de ganho de realimentação ótima pode ser calculada como:

$$K = (B^T PB + R)^{-1} B^T PA \tag{5.13}$$

onde P é uma matriz simétrica, definida não-negativa que pode ser obtida através da solução da Equação Algébrica de Riccati Discreta *EARD*:

$$A^T PA - P - A^T PB(R + B^T PB)^{-1} B^T PA + Q = 0 \tag{5.14}$$

A, Q e R são matrizes reais quadradas com Q simétrica e A, R não-singulares, [143, 160].

5.3 Equação Dinâmica Neuro-LMI-Riccati Discreta

Uma propriedade muito útil de uma *LMI* é a de poder converter desigualdades não-lineares em desigualdades lineares, usando o complemento de Schur, visto no Lema2.

Nesta seção, vamos apresentar nossa proposta [150] para resolver a seguinte *IARD*:

$$A^T PA - P + A^T PB(-R - B^T PB)^{-1} B^T PA + Q < 0 \tag{5.15}$$

construindo a respectiva *LMI* e usando uma abordagem com *RNR*. Para fazer isto propomos o seguinte Teorema:

Teorema 2 *As seguintes inequações são equivalentes:*

$$A^T PA - P + A^T PB(-R - B^T PB)^{-1} B^T PA + Q < 0 \tag{5.16}$$

e

$$\tilde{A}^T P\tilde{B}^T + \tilde{B}P\tilde{A} + \tilde{C} - \tilde{B}P\tilde{B}^T - \tilde{B}1P\tilde{B}1^T < 0 \tag{5.17}$$

onde os termos são definidos durante o desenvolvimento da prova.

Prova 2 *Escrevendo a inequação matricial correspondente à equação (5.16), tem-se:*

$$\begin{bmatrix} A^T PA - P + Q & A^T PB \\ B^T PA & R + B^T PB \end{bmatrix} < 0 \tag{5.18}$$

Esta inequação relacionada com a inequação de Riccati Discreta pode ser reescrita de forma a ser possível aplicar condições de solucionabilidade e estabilidade e obter uma LMI em P. Observa-se

que o primeiro elemento de (5.18): $A^T PA - P + Q$, corresponde à inequação de Lyapunov discreta; aplicando a ele o Lema2 resulta:

$$\begin{bmatrix} -P+Q & A^T P \\ PA & -P \end{bmatrix} < 0 \tag{5.19}$$

Repetindo o procedimento acima com o elemento $B^T PB + R$, a seguinte LMI é obtida:

$$\begin{bmatrix} -P & PB \\ B^T P & R \end{bmatrix} < 0 \tag{5.20}$$

Logo (5.18) pode ser reescrita como a seguinte LMI:

$$\begin{bmatrix} -P+Q & A^T P & 0 \\ PA & -P & PB \\ 0 & B^T P & R \end{bmatrix} < 0 \tag{5.21}$$

Definindo

$$\tilde{A} = \begin{bmatrix} A & 0 & B \end{bmatrix} \qquad \tilde{B} = \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix} \quad \tilde{C} = \begin{bmatrix} Q & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & R \end{bmatrix} \qquad \tilde{B}1 = \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$$

A LMI (5.21) corresponde à expressão (5.17).

Para resolver a LMI em termos de uma *RNR*, impomos duas matrizes de folga, $\widetilde{R_1}, \widetilde{R_2}$, as quais convertem a LMI (5.17) no seguinte sistema de equações matriciais lineares:

$$\begin{aligned} G(P, R_1) &= \tilde{A}^T P \tilde{B}^T + \tilde{B} P \tilde{A} + \tilde{C} - \tilde{B} P \tilde{B}^T - \tilde{B}1 P \tilde{B}1^T + \widetilde{R_1}\widetilde{R_1}^T = 0 \\ H(P, R_2) &= P - \widetilde{R_2}\widetilde{R_2}^T = 0 \end{aligned} \tag{5.22}$$

As matrizes de folga $\widetilde{R_1}$ e $\widetilde{R_2}$ são restritas a serem definidas positivas e não-singulares, com a seguinte forma:

$$\widetilde{R_i} = \begin{bmatrix} h_{i1}(r_{i,11}) & 0 & 0 & \dots & 0 \\ r_{i,21} & h_{i2}(r_{i,22}) & 0 & \dots & 0 \\ r_{i,31} & r_{i,32} & h_{i3}(r_{i,33}) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ r_{i,n1} & r_{i,n2} & r_{i,n3} & \dots & h_{in}(r_{i,nn}) \end{bmatrix} \tag{5.23}$$

com $\widetilde{r}_{s,jj} = h_{sj}(r_{s,jj}) > 0, \qquad i, s = 1, 2, j = 1, \dots, n$

$$h_{sj}(r_{s,jj}) = \epsilon + \frac{p_{s,max} - \epsilon}{1 + e^{-\lambda_{r_{s,jj}}}}, \qquad \forall s, j \tag{5.24}$$

O nível do limitador $p_{s,max}$ é ajustável. Dependendo do valor do ajuste $p_{s,max}$, a rede neural poderá oferecer diferentes soluções para a *LMI* (5.16).

Para determinar as equações dinâmicas neurais, um novo problema de otimização é definido com a função objetivo (5.25), e com as funções $G(P, R_1)$ e $H(P, R_2)$ definidas em (5.22):

$$\min E[GH(P, R_1, R_2)] \quad = \sum_{i=1}^{n}\sum_{j=1}^{n} e_{1,ij}[G(P, R_1)] + e_{2,ij}[H(P, R_2)] \tag{5.25}$$

Aplicando novamente as equações dinâmicas gerais expostas em (4.10-4.11), no Capítulo 4 e as propriedades de derivação lá apresentadas no Lema 1, obtemos as equações dinâmicas discretas que resolvem a IARD com uma *RNR*:

$$\begin{array}{ll} \frac{dV}{dt} & = -\eta_v[\tilde{A}F_1(P,\widetilde{R_1})\tilde{B} + \tilde{B}^T F_1(P,\widetilde{R_1})\tilde{A}^T - \tilde{B}^T F_1(P,\widetilde{R_1})\tilde{B} - \tilde{B}1^T F_1(P,\widetilde{R_1})\tilde{B}1 + F_2(P,\widetilde{R_2})] \\ \frac{dR_1}{dt} & = -\eta_{r_1} F_1(P,\widetilde{R_1})\widetilde{R_1} \\ \frac{dR_2}{dt} & = -\eta_{r_2} F_2(P,\widetilde{R_2})\widetilde{R_2} \end{array} \tag{5.26}$$

onde as matrizes de ativação são definidas por

$$\begin{array}{ll} F_1(P,\widetilde{R_1}) & = F_1(\tilde{A}^T P\tilde{B}^T + \tilde{B}P\tilde{A} + \tilde{C} - \tilde{B}P\tilde{B}^T - \tilde{B}1P\tilde{B}1^T + \widetilde{R_1}\widetilde{R_1}^T) \\ F_2(P,\widetilde{R_2}) & = F_2(P - \widetilde{R_2}\widetilde{R_2}^T) \end{array}$$

5.4 Estrutura da Rede Neural

A arquitetura da *RNR* proposta para resolver o problema *Neuro-LMI-Riccati*, está constituída de cinco camadas conectadas bidirecionalmente. A camada 1 (ou primeira) é de entrada, representada por $F_1(P,\widetilde{R_1}) = [f1_{ij}(t)]$; as três camadas intermédiarias representadas por $F_2(P,\widetilde{R_2}) = [f2_{ij}(t)]$, $\widetilde{R_1(t)} = [r_{1,ij}(t)]$ e $\widetilde{R_2(t)} = [r_{2,ij}(t)]$, respectivamente, são conetadas bidirecionalmente com a camada de saída; e finalmente uma camada de saída, representada por $V(t) = [v_{ij}(t)]$. A partir disso, podemos fazer P corresponder à saída final da rede ($v_{ij}(t)$). A matriz de ativação de estado $V(t)$ para os neurônios na camada de saída representa o resultado computacional de P (solução da inequação (5.16)) e as matrizes de estado $R_1(t), R_2(t)$ representam as soluções para as matrizes de folga, $\widetilde{R_1}, \widetilde{R_2}$ respectivamente. As camadas para $V(t), F_1(P,\widetilde{R_1}), F_2(P,\widetilde{R_2})$ vão consistir de arranjos quadrados $(n \times n)$ de neurônios.

5.5 LMI-Neural de Riccati Contínua

Nesta seção apresentamos as equações dinâmicas neurais que descrevem a equação de Riccati contínua utilizando *RNR*; detalhes deste procedimento estão em [87].

Para motivação considere um sistema linear invariante e contínuo no tempo completamente controlável:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{5.27}$$

Analogamente, deseja-se minimizar a função de custo quadrática

$$J = \int_0^\infty (x^T(t)Qx(t) + u^T(t)Ru(t))dt \tag{5.28}$$

A lei de controle resultante é: $u(t) = -Kx(t)$, que depende de uma matriz de ganho, K, definida através de uma matriz P correspondente à solução da inequação matricial algébrica de Riccati Contínua (*EARC*):

$$A^T P + PA - PSP + Q < 0 \tag{5.29}$$

onde $S = BR^{-1}B^T$ é uma matriz simétrica e semidefinida positiva.

Para fazer a formulação utilizando uma *RNR* na solução da equação (5.29), apresentamos apenas as equações semelhantes às definidas anteriormente para o caso discreto.

Para a função objetivo definida por:

$$\begin{array}{rcl} \min E[GH(P,R_1,R_2)] & = & \sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}[G(P,R_1)] + e_{ij}[H(P,R_2)] \\ & onde & \\ G(P,R_1) & = & \tilde{A}P\tilde{B} + \tilde{B}^T P\tilde{A}^T + \tilde{C} + \widetilde{R_1}\widetilde{R_1}^T \\ H(P,R_2) & = & P - \widetilde{R_2}\widetilde{R_2}^T \end{array} \tag{5.30}$$

as funções de ativação são definidas como:

$$\begin{array}{rcl} F_1(P,\widetilde{R_1}) & = & F(\tilde{A}P\tilde{B} + \tilde{B}^T P\tilde{A}^T + \tilde{C} + \widetilde{R_1}\widetilde{R_1}^T) \\ F_2(P,\widetilde{R_2}) & = & F(P - \widetilde{R_2}\widetilde{R_2}^T) \end{array}$$

As equações dinâmicas neurais que descrevem a *IARC* utilizando uma *RNR* são as seguintes:

$$\begin{array}{l} \frac{dV}{dt} = -\eta_v[\tilde{A}^T F_1(P,\widetilde{R_1})\tilde{B}^T + \tilde{B}F_1(P,\widetilde{R_1})\tilde{A} + F_2(P,\widetilde{R_2}) \\ \frac{dR_1}{dt} = -\eta_{r_1} F_1(P,\widetilde{R_1})\widetilde{R_1} \\ \frac{dR_2}{dt} = -\eta_{r_2} F_2(P,\widetilde{R_2})\widetilde{R_2} \end{array} \tag{5.31}$$

5.6 Implementações e Resultados das Neuro-LMI-Riccati

Nesta seção apresentamos os resultados das implementações neurais das inequações algébricas de Riccati discreta (*IARD*) e contínua (*IARC*) no tempo com vários exemplos. As implementações para resolver a **IARD Neural** e a **IARC Neural** em forma geral, foram desenvolvidas no Matlab usando o método de Runge-Kutta de quarta ordem explicado no Capítulo anterior; neste caso as equações diferenciais coincidem com as equações dinâmicas neurais obtidas em cada problema analisado: Discreto e Contínuo, respectivamente. A estrutura do algoritmo para calcular $V(t)$, por exemplo, no caso discreto é:

s_1 = feval('fv',etav,V,U,A,B,B1,Y);

s_2 = feval('fv',etav,V+h*s_1/2,A,B,B1,Y);

s_3 = feval('fv',etav,V+h*s_2/2,A,B,B1,Y);

s_4 = feval('fv',etav,V+h*s_3,S,A,B,B1,Y);

V = V + h*$(s_1 + 2 * s_2 + 2 * s_3 + s_4)$/6;

Cada s_i corresponde a uma variável que é encarregada da atualização de um peso sináptico da rede ou equivalentemente à variável desconhecida do problema, x.

A seguir apresentamos um exemplo utilizando uma inequação de Riccati discreta no tempo. Vale resaltar que este conjunto de inequações é difícil de resolver por ter termos quadráticos na sua estrutura. Apresentamos uma primeira versão do algoritmo com o qual obtemos bons resultados.

Exemplo 1. Este exemplo apresenta um problema de controle linear quadrático discreto no tempo definido por (5.10-5.16). O desempenho da solução da *IARD*, usando uma abordagem de rede neural recorrente multicamada é apresentado. Os parâmetros para as matrizes do sistema são:

$$A = \begin{bmatrix} 0.9512 & 0 \\ 0 & 0.9048 \end{bmatrix} \quad B = \begin{bmatrix} 4.877 & 4.877 \\ -1.1895 & 3.569 \end{bmatrix} \quad R = \begin{bmatrix} 0.33 & 0 \\ 0 & 3 \end{bmatrix} \quad Q = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.02 \end{bmatrix}$$

As equações dinâmicas neurais (5.26) foram utilizadas para resolver a inequação de Riccati Discreta (5.16). As soluções de regime permanente de V e $\widetilde{R}_1$ obtidas para a respectiva **IARD Neural** são:

$$V = \begin{bmatrix} 0.0198 & -0.0069 \\ -0.0069 & 0.0951 \end{bmatrix}; \quad \widetilde{R1} = \begin{bmatrix} 0.6497 & 0 & 0 & 0 & 0 & 0 \\ 0.1286 & 0.6573 & 0 & 0 & 0 & 0 \\ 0.0045 & 0.0189 & 0.6541 & 0 & 0 & 0 \\ -0.0146 & -0.0482 & 0.2374 & 0.6716 & 0 & 0 \\ 0.0326 & 0.0303 & -0.0316 & 0.0456 & 0.6257 & 0 \\ 0.0012 & 0.0025 & -0.0620 & -0.0914 & 0.0040 & 0.4881 \end{bmatrix} \tag{5.32}$$

Esta matriz V é simétrica, definida positiva e satisfaz a IARD (5.16). A simulação foi realizada usando os seguintes parâmetros, $f_{kl}(\xi) = \xi$, $(\eta_v, \eta_{r1}, \eta_{r2}) = (0.1, 1, 2)$, $p_{s,max} = 0.5$, $\epsilon = 0.01$, $\Delta t = 10^{-5}$, $P(0) = [0.1]_{2\times 2}$, $x(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ e $Y(0) = I$. Utilizamos as equações dinâmicas neurais apresentadas em (5.26) para resolver a inequação matricial algebrica de Riccati discreta apresentada em (5.16), em toda a faixa de operação.

Os resultados das simulações em tempo real são apresentadas na Fig.5.1, onde respectivamente ilustra-se o comportamento convergente da matriz de estado $V(t)$ e da matriz de folga $\widetilde{R1}$. A partir da Fig.5.1, podemos observar que os elementos de $V(t)$ atingem seu valor de regime permanente rapidamente.

Portanto, nossa proposta e implementação para resolver a **IARD** utilizando *RNR*, apresenta bons resultados que podem ser testados e comprovados com qualquer outro método de solução, em especial [22, 143].

Exemplo 2 Neste segundo exemplo, só fizemos variações nas taxas de aprendizado da rede neural do exemplo acima. Os valores utilizados foram os seguintes, $P(0) = [0.1]_{2x2}$ and $x(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $(\eta_v, \eta_{r1}, \eta_{r2}) = (5, 10.5, 15.5)$, $p_{s,max} = 0.5$ e $\epsilon = 0.01$. Utilizamos, novamente as equações dinâmicas neurais apresentadas em (5.26) para resolver a inequação matricial algebrica de Riccati discreta apresentada em (5.16), em cada faixa de operação.

Os resultados das simulações são apresentados na Figs.5.2 e 5.4, onde respectivamente ilustra-se o comportamento convergente da matriz solução V, da matriz $\widetilde{R1}$ e finalmente da matriz $G(P, R_1)$.

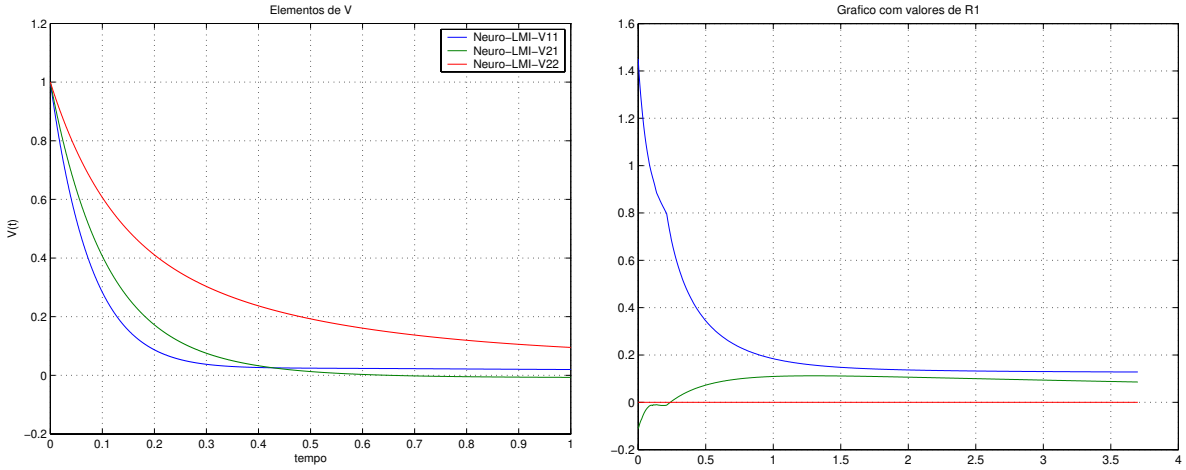

Fig. 5.1: Trajetórias de V e $\widetilde{R_1}$, respectivamente

As matrizes resultantes serão apresentadas a seguir.

$$V = \begin{bmatrix} 0.0019 & -0.0041 \\ -0.0041 & 0.0112 \end{bmatrix}; \quad \widetilde{R_1} = \begin{bmatrix} 0.1062 & 0 & 0 & 0 & 0 & 0 \\ 0.1272 & 0.1042 & 0 & 0 & 0 & 0 \\ 0.1932 & 0.1104 & 0.0986 & 0 & 0 & 0 \\ -0.0192 & -0.0004 & 0.1393 & 0.2116 & 0 & 0 \\ -0.0186 & 0.0274 & 0.0060 & 0.0692 & 0.0357 & 0 \\ 0.0005 & 0.0002 & -0.0007 & -0.0014 & 0.0000 & 0.0000 \end{bmatrix}; \tag{5.33}$$

Esta matriz V é simétrica, definida positiva e satisfaz a LMI (5.17).

Exemplo 3.Considere o controle ótimo de um sistema contínuo no tempo, estável em malha aberta, descrito por (5.27) e (5.28), para o qual são dadas as matrizes:

$$A = \begin{bmatrix} -4 & 1 \\ 0 & -7 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

As equações dinâmicas neurais (5.31) foram utilizadas para resolver a inequação de Riccati Contínua (5.29). As soluções de regime permanente de V e $\widetilde{R_1}$ obtidas para a respectiva **IARC Neural** são:

$$V = \begin{bmatrix} 0.1388 & 0.0283 \\ 0.0283 & 0.2378 \end{bmatrix}; \quad \widetilde{R_1} = \begin{bmatrix} 0.3295 & 0 & 0 & 0 \\ 0.5227 & -0.0003 & 0 & 0 \\ -0.5083 & 0.7095 & 0.4880 & 0 \\ -0.5089 & -0.5576 & 0.2805 & 0.5928 \end{bmatrix}; \tag{5.34}$$

Esta matriz V é simétrica, definida positiva e satisfaz a IARC (5.29). A simulação foi realizada usando os seguintes parâmetros, $f_{kl}(\xi) = \xi$, $(\eta_v, \eta_{r1}, \eta_{r2}) = (0.1, 100, 100)$, $p_{s,max} = 1$, $\epsilon = 0.01$, $\Delta t = 10^{-5}$, $P(0) = [0.1]_{2\times 2}$, $x(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ e $Y(0) = I$. Utilisamos as equações dinâmicas neurais apresentadas em (5.31) para resolver a inequação matricial algebrica de Riccati contínua apresentada em (5.29), em toda a faixa de operação.

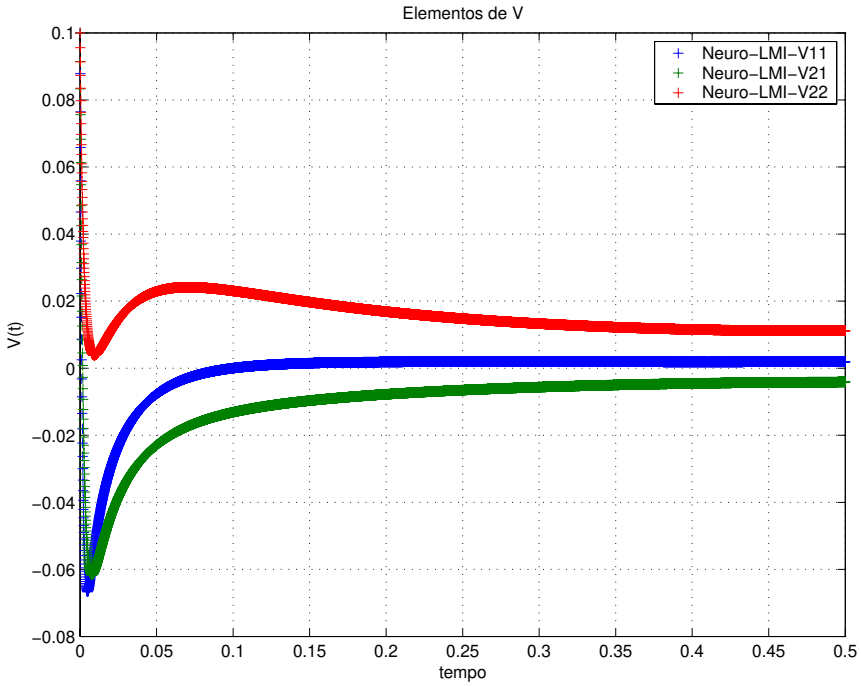

Fig. 5.2: Trajetórias de V

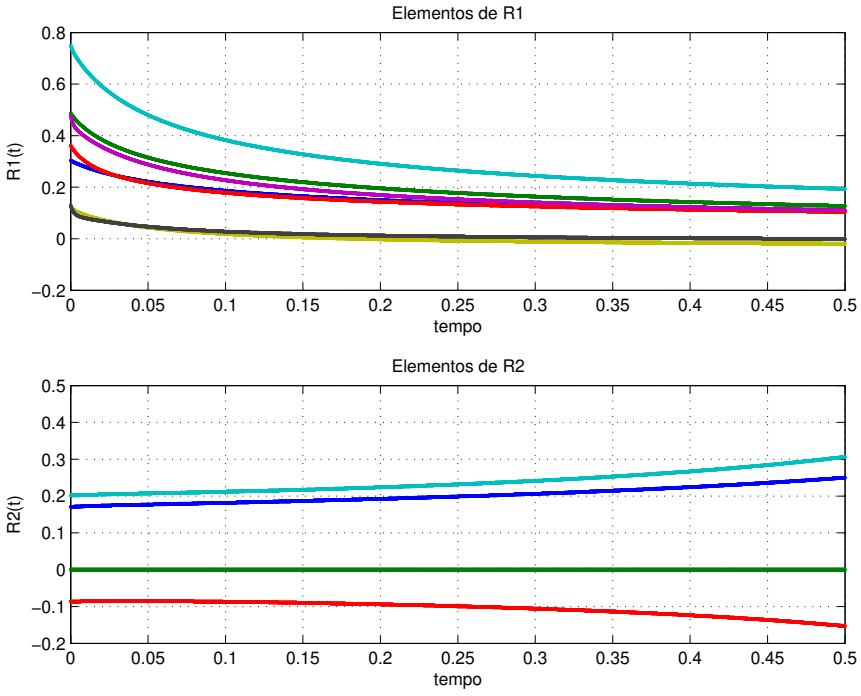

Fig. 5.3: Trajetórias de $\widetilde{R_1}$

Os resultados das simulações são apresentados na Fig.5.5, onde respectivamente ilustra-se o comportamento convergente da matriz de estado V e da matriz de folga $\widetilde{R_1}$. A partir da Fig. 5.5, podemos

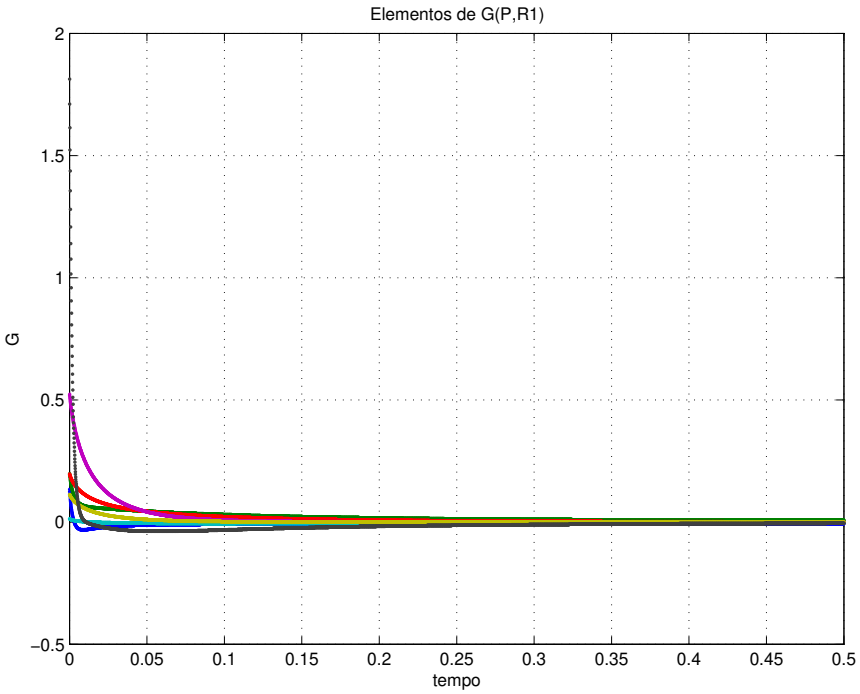

Fig. 5.4: Trajetória de $G(P, R_1)$

observar que os elementos de $V(t)$ atingem seu valor de regime permanente rapidamente.

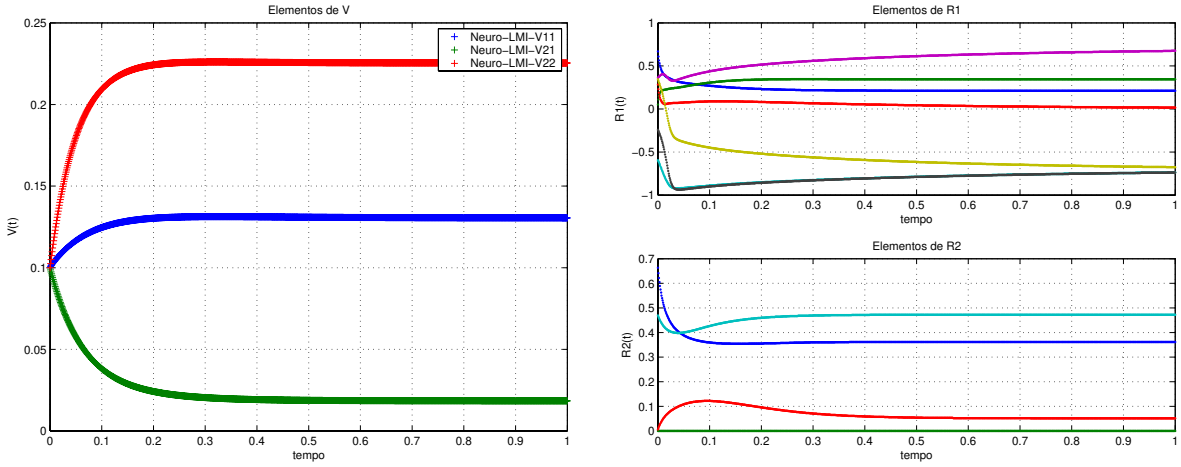

Fig. 5.5: Trajetórias de V e $\widetilde{R1}$.

Capítulo 6

Identificação e Controle Inteligente no Espaço de Estado

6.1 Introdução

Neste Capítulo propomos um esquema de controle adaptativo com Escalonamento de Ganhos baseado em Redes Neurais para sistemas multivariáveis discretos variantes no tempo. O algoritmo **MOESP_VAR**, proposto no Capítulo 3, é utilizado para a modelagem computacional da planta a partir de dados de entrada-saída, de forma a obter modelos lineares multivariáveis discretos invariantes no tempo em vários pontos de operação. Uma lei de controle linear quadrática seguidor ótima é desenvolvida em malha fechada para cada modelo multivariável identificado, tal que o sistema acompanha uma trajetória desejada num intervalo de tempo dado. Uma abordagem alternativa incluíndo incertezas aditivas aleatórias na dinâmica dos modelos identificados é proposta, resultando num controlador suficientemente robusto para estabilizar a planta variante no tempo. Um escalonador de ganhos neural é projetado via algoritmo *backpropagation*, para ajustar *on-line* os controladores ótimos projetados. Em síntese, propomos e implementamos uma estrutura de controle inteligente para sistemas discretos multivariáveis variantes no tempo através de uma abordagem que pode ser chamada **ILPV** (Intelligent Linear Parameter Varying). Por meio dela concretizamos um controlador LPV Inteligente. Resultados de simulações demonstram a eficiência da metodologia proposta para uma planta multivariável com autovalores variantes no tempo, importante, por exemplo, em aplicações em controle de aeronaves e de manipuladores robóticos.

Os conceitos e métodos desenvolvidos em *Teoria de Sistemas de Controle* e as novas técnicas em desenvolvimento no campo da *Inteligência Computacional* [9] como as *Redes Neurais* e os *Algoritmos Genéticos* podem ser convenientemente combinados para o Controle Inteligente de Sistemas Dinâmicos Complexos na presença de incertezas. Estudos de Narendra [105, 103] mostram que a utilização das *Redes Neurais* pode fornecer melhores soluções que as técnicas tradicionais de identificação e Controle. Seus trabalhos podem ser considerados os primeiros passos na identificação e controle de sistemas utilizando *Redes Neurais*. Na visão de Harris, [54] pouco ganho se obtém quando aplica-se Controle Inteligente a sistemas lineares invariantes no tempo. O mesmo não se deve dizer para o caso de sistemas lineares variantes no tempo, tema deste estudo.

Intrinsicamente relacionado ao problema de identificação está o problema de *Controle Adaptativo*, cuja motivação principal é muito atraente: um controlador que modifica-se baseado no compor-

tamento da planta controlada, de forma a satisfazer à algumas especificações de projeto [64, 8]. O elemento principal deste método de projeto de controladores é o mecanismo de ajuste dos parâmetros do controlador. Entre os tipos principais de técnicas de ajuste destes parâmetros podemos mencionar o Escalonamento de Ganhos (EG), o controle adaptativo baseado num modelo de referência, etc, [111].

O Controle Adaptativo é uma importante área de pesquisa de muitos pesquisadores, entre os quais podemos citar [8, 77, 91, 98, 102] e tem sido utilizado principalmente para melhorar o desempenho on-line dos controladores. O desenvolvimento da teoria de controle adaptativo e sua viabilização em microprocessadores conduziu a uma série de aplicações com bons desempenhos em áreas tais como robótica, controle de aeronaves, comunicação e reatores, dentre outras.

A dificuldade em projetar, por exemplo, uma lei de controle de vôo está em como obter um controlador que possa ajustar-se às mudanças do modelo dinâmico de uma aeronave garantindo que um dado desempenho seja alcançado. Dentre os muitos métodos para projetar controladores temos: controle PID, controle com Escalonamento de Ganhos, [168] e controle inteligente.

Controle Inteligente foi originalmente proposto por Fu, [50], e foi definido como uma abordagem para gerar ações de controle pelo emprego de aspectos de inteligência artificial, pesquisa operacional e sistemas de controle automático. É uma área de aplicação de Inteligência Artificial ao Controle de Sistemas considerada sucessora do controle adaptativo da década de 1970. Estratégias são definidas e buscam ser capazes de alcançar e manter o nível desejado de desempenho na presença de grandes incertezas em sistemas de malha fechada. No controle de sistemas complexos, dentre as dificuldades que aparecem, ressaltaremos as que podem ser classificadas em quatro categorias: complexidade computacional, não-estacionariedade, não-linearidade e incerteza. Aos sistemas de controle capazes de lidar com tais categorias de dificuldades utilizando inteligência computacional chamaremos *Sistemas de Controle Inteligente*.

O uso da terminologia *Controle Inteligente* agrupa diversas metodologias, [21, 89], combinando a teoria de controle convencional com técnicas de inteligência computacional baseadas em redes neurais, lógica nebulosa, sistemas especialistas, algoritmos genéticos e uma ampla variedade de técnicas de busca e otimização.

Em [130, 131] os autores apresentam um controle inteligente hierárquico, que inclui uma Rede Neural e a propriedade de aproximação de funções para manipuladores robóticos.

Sistemas de controle usando Redes Neurais e/ou Lógica Nebulosa são alternativas viáveis em controle adaptativo. Pesquisas com Redes Neurais para aplicações em controle estão sendo realizadas por alguns pesquisadores, [36, 103, 117, 129]. Alguns trabalhos no projeto de controladores neurais discretos no tempo foram realizados por [85, 86, 15, 128]; até o momento não se tem muitos resultados para o controle em malha fechada de sistemas não-lineares e/ou não-estacionários discretos no tempo usando Redes Neurais multicamadas.

Num sistema de controle realimentado medem-se as saídas do sistema, comparam-se os resultados com as saídas desejadas e então o sinal de erro produzido é utilizado para calcular as entradas de controle do sistema de tal maneira que o erro torne-se pequeno. Sistemas de controle realimentado produzidos pelo homem são responsáveis, por exemplo, pelos avanços que a era aeroespacial tem nos dias de hoje e também são usados no controle industrial, automotivo, etc.

Neste capítulo são considerados sistemas cujos modelos matemáticos são lineares porém ocorre variação nos seus parâmetros durante suas operações. Desse modo, um único controlador fixo não apresenta a necessária capacidade para manter o desempenho especificado em toda a faixa de operação

do sistema. Torna-se necessário um controlador que também varie com o sistema, visando manter o desempenho especificado. Admite-se que um ou mais parâmetros ou coeficientes da planta estejam sofrendo uma variação e que essa variação possa ser estimada em tempo real através da medição de um conjunto de variáveis denominada *vetor de parâmetros*; através dele será ajustado o controlador a ser aplicado em dados instantes. Se certos parâmetros do controlador são adequadamente corrigidos tendo em consideração dados pontos de operação, o método de controle é chamado *Gain Scheduling (GS)* ou *Escalonamento de Ganhos (EG)*. A Figura 6.1 apresenta um Sistema de Controle Adaptativo Clássico com EG contendo dois ciclos: um ciclo de realimentação composto pela planta mais o controlador e outro ciclo que faz o ajuste dos parâmetros do controlador baseado no conhecimento a priori que se tem dos pontos de operação da planta.

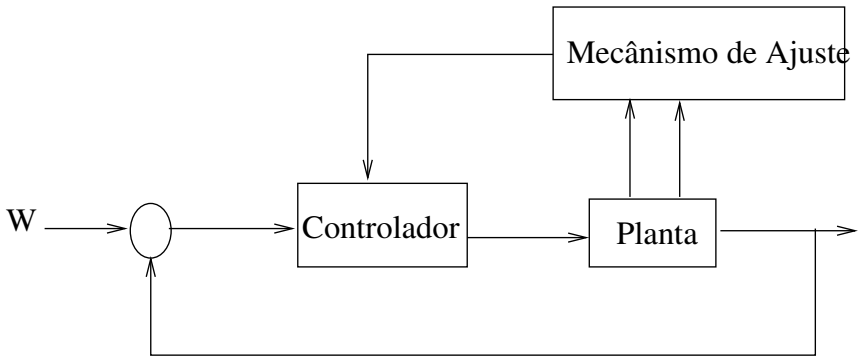

Fig. 6.1: Sistema de Controle Adaptativo Clássico por EG

O comportamento dinâmico do sistema a ser controlado muda com a região de operação. Por exemplo, sistemas não-lineares reais, comportam-se como lineares em regiões limitadas do espaço de estado. A abordagem clássica para controlar tais sistemas é linearizar os sistemas em alguns pontos de operação e projetar um ou mais controladores lineares para o sistema nestes pontos de operação. Os paradigmas modernos de controle tais como controle robusto H_∞ e controle adaptativo tratam disto requerendo um modelo nominal linear no espaço de estado mais algum modelo residual para realizar o projeto do controlador. Uma abordagem típica para esta situação é a que aplica a técnica de *Escalonamento de Ganhos*.

Ela tem se mostrado de grande utilidade em muitas aplicações de engenharia, por exemplo no controle de sistemas não-lineares bem comportados e no de sistemas lineares com parâmetros variantes no tempo (*LPV*), [13, 99, 3, 112]. Problemas com Escalonamento de Ganhos são objeto de importantes pesquisas tanto do ponto de vista teórico como do prático [73, 127].

6.2 Controle Inteligente com EG de Sistema LPV

Combinando a ideia básica de escalonamento de ganhos com o problema de controle adaptativo baseado em modelo, neste trabalho estamos interessados em propor um esquema de controle seguidor ótimo inteligente com *EG* usando Rede Neural Perceptron com múltiplas camadas (MLP), [138], para sistemas multivariáveis discretos variantes no tempo baseado em modelo identificado pelo método

MOESP_VAR proposto no Capítulo 3. A estrutura do controlador e os modelos selecionados são apresentados no desenvolvimento do capítulo.

No esquema adaptativo proposto, a estimativa dos parâmetros escalonados desconhecidos é obtida sucessivamente através do método proposto e implementado em [151], e estes alimentam o algoritmo de projeto que gera os controladores para estabilizar a planta e garantir um desempenho adequado deste sistema de controle inteligente com escalonamento neural de ganhos. O método de identificação MOESP_VAR, é utilizado para fazer a modelagem off-line da planta a partir de dados de entrada-saída, e determina modelos lineares discretos multivariáveis invariantes no tempo em vários pontos de operação para uma planta variante no tempo multivariável. Uma lei de controle seguidor linear quadrático ótimo em malha fechada com escalonamento de ganhos é desenvolvida, na seção 6.2.1, tal que para cada modelo linear multivariável identificado o sistema LPV acompanhe uma trajetória desejada num intervalo de tempo dado e de forma inteligente tenha um desempenho adequado para toda a região de operação.

Para efeito de projeto, uma abordagem alternativa incluíndo incertezas aditivas aleatórias na dinâmica dos modelos identificados é proposta, através da transformação dada na equação (6.12) resultando num controlador suficientemente robusto para estabilizar a planta variante no tempo.

Considere um sistema linear LPV, descrito pelas equações

$$\begin{cases} x_{k+1} & = A(\sigma)x_k + B(\sigma)u_k \\ y_k & = C(\sigma)x_k \end{cases} \tag{6.1}$$

com $x_k \in \Re^n$, $u_k \in \Re^m$, $y_k \in \Re^l$. O projeto com a metodologia de escalonamento de ganhos consiste em selecionar um subconjunto finito de N pontos $\{\sigma = 1, 2, 3, \ldots, N\}$ representativos para a dinâmica da planta, do conjunto I de pontos de operação σ, e projetar um controlador linear invariante no tempo para cada valor naquele subconjunto de pontos de operação. O projeto é tal que num ponto de operação o sistema de malha fechada tenha as propriedades especificadas de desempenho para o mesmo.

São utilizadas Redes Neurais MLP para a implementação de controlador com *escalonamento de ganhos* para planta linear variante no tempo MIMO com lei de controle seguidor ótimo. São selecionados diversos pontos de operação e as respectivas referências ($r(\sigma)$) de modo a cobrir toda a faixa de operação da planta, Para cada um destes pontos de operação, obtemos através do método de identificação $MOESP\_VAR$, um modelo linear identificado multivariável invariante no tempo. A partir destes são projetados os controladores lineares correspondentes para um número finito N de pontos de operação pertencendo ao conjunto I. Logo vamos ter N modelos lineares identificados invariantes no tempo, para um sistema variante no tempo linear original dado, operando em N pontos especificados. Uma Rede Neural é então treinada para realizar um controlador inteligente de modo a substituir todos os controladores lineares.

Os controladores lineares assim obtidos, para cada ponto de operação $r(\sigma), \sigma \in I$, são então interpolados através de uma Rede Neural MLP para cobrir todo o conjunto I.

As RNA têm sido bastante estudadas e suas propriedades de aprendizagem, adaptabilidade, classificação, aproximação de funções e outras têm sido muito utilizadas nas aplicações em processamento de sinais e identificação de sistemas em malha aberta. Já nas aplicações das Redes Neurais em sistemas de controle em malha fechada de sistemas dinâmicos lineares e não-lineares, elas são de grande importância devido às suas características de mapeadores universais e à sua capacidade de aprender por treinamento, [39, 15, 114, 59].

Elas têm sido satisfatoriamente aplicadas como ferramenta de aproximação em sistemas dinâmicos e estáticos. Em [48] foi provado que qualquer mapeamento contínuo pode ser realizado utilizando uma RNA multicamada com até uma camada escondida usando uma função de ativação sigmoidal; [52] propõe a utilização de uma rede função de base radial (RBF) quando uma única aproximação for necessária e em [60] mostra-se que a RNA *feedforward* multicamadas usando função de ativação: linear, hiperbólica, sigmoide, etc, é capaz de aproximar qualquer função mensurável com qualquer grau de precisão.

Os controladores neurais constituem sistemas de aprendizado adaptativo. Um fato importante é que controladores neurais podem apresentar melhores resultados que controladores adaptativos convencionais lineares e não-lineares quanto aos aspectos de precisão e robustez de sistemas em malha fechada.

O procedimento para a obtenção do controlador neural com escalonamento de ganhos proposto é desenvolvido nas etapas seguintes:

- Determinar os diversos pontos de operação para a planta (6.1).
- Obter modelos lineares invariantes no tempo para diversos pontos de operação da planta variante no tempo, através do algoritmo $MOESP\_VAR$.
- Projetar controladores lineares baseados nos modelos lineares obtidos em cada ponto de operação, pela proposta de controle seguidor ótimo apresentada na seção 6.2.1.
- Treinar via *backpropagation* um controlador neural com escalonamento de ganhos em substituição aos controladores lineares obtidos nos pontos de operação correspondentes.
- Projetar um controlador inteligente com escalonamento de ganhos para ajustar e interpolar online os controladores ótimos projetados acima.

O controlador ILPV assim projetado via algoritmo *backpropagation*, tem as seguintes propriedades:

- Melhora o desempenho da malha fechada;
- Trabalha com um adaptador inteligente;
- Atualiza todos os controladores para o sistema LPV;
- Fornece uma suave interpolação entre os controladores nos diversos pontos de operação para o sistema LPV.

6.2.1 Controlador Seguidor LPV Ótimo

De posse do modelo identificado da planta linear invariante no tempo obtido com o algoritmo $MOESP\_VAR$, para um ponto de operação em um σ-ésimo instante, queremos determinar um esquema de controle seguidor ótimo para um sistema discreto no tempo, ou seja, estamos interessados em fazer que a saída acompanhe um sinal de referência conhecido desejado $r(\sigma)$, num intervalo de tempo predefinido, usando uma lei de controle seguidor ótimo em malha fechada.

Desejamos, por exemplo projetar um piloto automático para uma aeronave ou para um veículo lançador de satélites ou controlar manipuladores robóticos, [134, 86]. Para fazer isto devemos minimizar o seguinte índice de desempenho quadrático com estado final fixo:

$$J(\sigma) = \frac{1}{2}(C(\sigma)x_N - r_N)^T L(C(\sigma)x_N - r_N) + \frac{1}{2}\sum_{k=0}^{N-1}[(C(\sigma)x_k - r_k)^T Q(C(\sigma)x_k - r_k) + u_k^T R u_k] \quad (6.2)$$

onde $L \geq 0, Q \geq 0$ e $R > 0$ são matrizes de ponderação conhecidas, simétricas e definidas positivas para um dado σ. A solução do problema de controle seguidor ótimo leva ao conjunto de equações, (6.5). Para cada valor de k temos um vetor de multiplicadores de Lagrange λ_{k+1} para um ponto de operação em um σ-ésimo instante. Reescrevendo conjuntamente (6.1) e (6.2) obtemos:

$$\hat{J}(\sigma) = \frac{1}{2}(C(\sigma)x_N - r_N)^T L(C(\sigma)x_N - r_N) + \frac{1}{2}\sum_{k=0}^{N-1}[(C(\sigma)x_k - r_k)^T Q(C(\sigma)x_k - r_k) + u_k^T R u_k] \quad (6.3)$$

$$+\lambda_{k+1}^T(-x_{k+1} + A(\sigma)x_k + B(\sigma)u_k) \quad (6.4)$$

A minimização de $\hat{J}(\sigma)$ em relação aos vetores λ_k, x_k, u_k para um σ-ésimo instante produz:

1. Equação de Estado:
$$x_{k+1} = A(\sigma)x_k + B(\sigma)u_k;$$

2. Equação de Coestado:
$$\lambda_k = A^T(\sigma)\lambda_{k+1} + C^T(\sigma)QC(\sigma)x_k - C^T(\sigma)Qr_k; \quad (6.5)$$

3. Condição de Estacionariedade:
$$0 = Ru_k + B^T(\sigma)\lambda_{k+1};$$

e as condições de contorno são dadas por

$$\lambda_N \quad = C^T(\sigma)L(C(\sigma)x_N - r_N), \qquad x_0 \quad \text{dado} \quad (6.6)$$

Da condição de estacionariedade, obtemos o controle ótimo para o σ-ésimo instante

$$u_k(\sigma) \equiv \breve{u}_k \quad = -R^{-1}B^T(\sigma)\breve{\lambda}_{k+1} \ ; \quad (6.7)$$

Escrevendo em forma compactada, temos o seguinte sistema Hamiltoniano para um dado σ:

$$\begin{bmatrix} x_{k+1} \\ \lambda_k \end{bmatrix}_\sigma = \begin{bmatrix} A(\sigma) & -B(\sigma)R^{-1}B^T(\sigma) \\ C^T(\sigma)QC(\sigma) & A^T(\sigma) \end{bmatrix} \begin{bmatrix} x_k \\ \lambda_{k+1} \end{bmatrix}_\sigma + \begin{bmatrix} 0 \\ -C^T(\sigma)Q \end{bmatrix} r(\sigma) \quad (6.8)$$

Podemos observar que esta lei de controle tem parte das condições de contorno num instante inicial e parte das condições de contorno num instante final. Para resolver este problema, vamos expressar $\breve{u}_k$ como uma combinação linear de variáveis de estado mais um termo dependente de $r(\sigma)$ baseados no método de Swap [31].

Admite-se que para todo $k \leq N$, $\breve{\lambda}_k$, para um σ-ésimo instante, possa ser escrito sob a forma:

$$\breve{\lambda}_k = \breve{P}_k \breve{x}_k - \breve{v}_k \tag{6.9}$$

para alguma seqüência auxiliar desconhecida $P_k \in \Re^{n \times n}$ e $v_k \in \Re^n$, no σ-ésimo instante.

Substituindo (6.9) em (6.7) e reescrevendo o modelo (6.1) temos:

$$\begin{array}{rl} \breve{x}_{k+1} & = A(\sigma)\breve{x}_k - B(\sigma)R^{-1}B^T(\sigma)\breve{P}_{k+1}\breve{x}_{k+1} + B(\sigma)R^{-1}B^T(\sigma)\breve{v}_{k+1} \\ \breve{x}_{k+1} & = (I + B(\sigma)R^{-1}B^T(\sigma)\breve{P}_{k+1})^{-1}(A(\sigma)\breve{x}_k + B(\sigma)R^{-1}B^T(\sigma)\breve{v}_{k+1}) \end{array} \tag{6.10}$$

Substituindo as equações (6.9) e (6.10) na equação de coestado obtemos:

$$\begin{array}{rl} \breve{P}_k\breve{x}_k - \breve{v}_k & = A(\sigma)^T\breve{P}_{k+1}[(I + B(\sigma)R^{-1}B^T(\sigma)\breve{P}_{k+1})^{-1}(A(\sigma)\breve{x}_k + B(\sigma)R^{-1}B^T(\sigma)\breve{v}_{k+1})] \\ & -A^T(\sigma)\breve{v}_{k+1} + C^T(\sigma)QC(\sigma)\breve{x}_k - C^T(\sigma)Q\breve{r}_k \end{array} \tag{6.11}$$

Resolvendo para $\breve{x}_k$ e $\breve{v}_k$ e utilizando o lema de inversão de matrizes, tem-se,

$$\begin{array}{rl} \breve{P}_k & = A^T(\sigma)[\breve{P}_{k+1} - \breve{P}_{k+1}B(\sigma)(B^T(\sigma)\breve{P}_{k+1}B(\sigma) + R)^{-1}B^T(\sigma)\breve{P}_{k+1}]A(\sigma) + C^T(\sigma)QC(\sigma) \\ \breve{v}_k & = [A(\sigma)^T - A^T(\sigma)\breve{P}_{k+1}B(\sigma)(B^T(\sigma)\breve{P}_{k+1}B(\sigma) + R)^{-1}B^T(\sigma)]\breve{v}_{k+1} + C^T(\sigma)Q\breve{r}_k \end{array} \tag{6.12}$$

onde $\breve{P}_k$ corresponde à solução da equação recorrente de Riccati discreta associada ao problema de controle seguidor num σ-ésimo instante.

Se comparamos as expressões (6.9) e (6.6) verificamos que podemos escrever:

$$\begin{array}{c} \breve{P}_N = \breve{C}^T L \breve{C} \\ \breve{v}_N = \breve{C}^T L \breve{r}_N \end{array} \tag{6.13}$$

Por conseguinte, a equação (6.12), pode ser resolvida de forma única, caminhando-se para trás, variando k de N até 0. Isto é, pode-se obter $\breve{P}_N, \breve{P}_{N-1} ... \breve{P}_0$, começando por $\breve{P}_N$, que é conhecido.

Tomando como referência a equação (6.9), o vetor de controle seguidor ótimo $\breve{u}_k$ pode ser expresso como,

$$\breve{u}_k = -R^{-1}B^T(\sigma)(\breve{P}_{k+1}\breve{x}_{k+1} - \breve{v}_{k+1}) \tag{6.14}$$

mas ainda não temos uma solução, pois o controle $\breve{u}_k$ agora depende do vetor de estado $\breve{x}_{k+1}$ o qual é desconhecido no instante k para o ponto de referência σ. Substituindo a equação de estado (6.5) em (6.14) e fazendo algumas manipulações algébricas, obtemos:

$$\begin{array}{rl} \breve{u}_k & = -R^{-1}B^T(\sigma)[\breve{P}_{k+1}(A(\sigma)\breve{x}_k + B(\sigma)\breve{u}_k) + \breve{v}_{k+1}] \\ & = (B^T(\sigma)\breve{P}_{k+1}B(\sigma) + R)^{-1}B^T(\sigma)(-\breve{P}_{k+1}A(\sigma)\breve{x}_k + \breve{v}_{k+1}) \end{array} \tag{6.15}$$

e então podemos definir dois ganhos para o controlador LPV:

$$\begin{array}{rl} \breve{K}_k & = (B^T(\sigma)\breve{P}_{k+1}B(\sigma) + R)^{-1}B^T(\sigma)\breve{P}_{k+1}A(\sigma) \\ \breve{K}_k^v & = (B^T(\sigma)\breve{P}_{k+1}B(\sigma) + R)^{-1}B^T(\sigma) \end{array} \tag{6.16}$$

Logo podemos expressar em forma geral a lei de controle $\breve{u}_k$, a matriz $\breve{P}_k$, o vetor $\breve{v}_k$ e a planta em malha fechada como

$$\begin{aligned} \breve{u}_k &= -\breve{K}_k\breve{x}_k + \breve{K}_k^v\breve{v}_{k+1} \\ \breve{P}_k &= A^T(\sigma)\breve{P}_{k+1}(A(\sigma) - B(\sigma)\breve{K}_k) + C^T(\sigma)QC(\sigma) \\ \breve{v}_k &= (A(\sigma) - B(\sigma)\breve{K}_k)^T\breve{v}_{k+1} + C^T(\sigma)Q\breve{r}_k \\ \breve{x}_{k+1} &= (A(\sigma) - B(\sigma)\breve{K}_k)\breve{x}_k + B(\sigma)\breve{K}_k^v\breve{v}_{k+1} \end{aligned} \tag{6.17}$$

O problema de projetar um sistema de controle seguidor ótimo minimizando um índice de desempenho quadrático para um dado σ, pode ser reduzido ao problema de obter uma solução recursiva definida positiva de uma equação de Riccati discreta $\breve{P}_k$ para aquele σ.

6.2.2 Escalonamento de Ganhos Neural

Nesta seção funções de aprendizagem da Rede Neural são utilizadas para projetar controles seguidores auto-ajustáveis concretizando um controlador inteligente com escalonamento de ganhos. A estrutura deste controlador ILPV como um todo é mostrada na Fig.6.2, e consiste em:

- Controlador com Escalonamento de Ganhos Neural (CEGN),
- Algoritmo de Controle Seguidor LPV Ótimo.

A arquitetura do controlador neural tem um total de quatro camadas, sendo uma camada de entrada com 2 neurônios cuja função de ativação é uma Tangente Hiperbólica Sigmoide (tansig) (um neurônio para cada linha da matriz r_σ), duas camadas escondidas com nove neurônios com função de ativação Linear (purelin) em cada camada oculta e finalmente uma camada de saída com função de ativação Linear(purelin) com os oito elementos correspondentes aos elementos das matrizes $\breve{K}_k$ e $\breve{K}_k^v$. A arquitetura do controlador ILPV é apresentada na Fig.6.2. A função de ativação de cada uma das camadas é introduzida a seguir. Funções de Ativação:

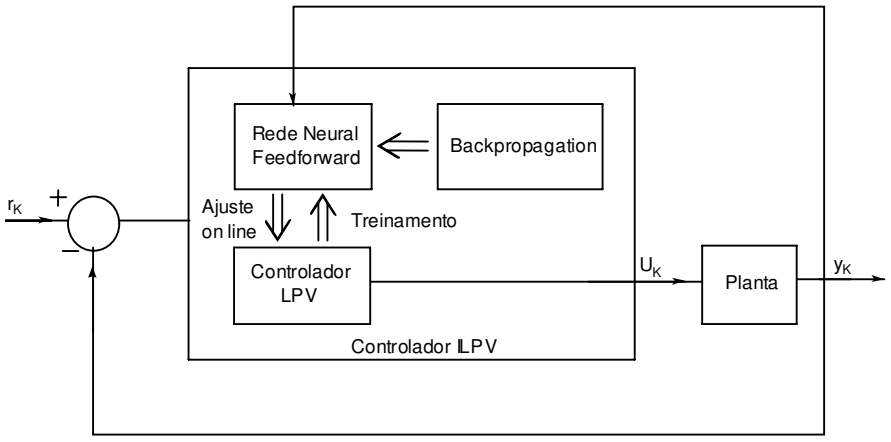

Fig. 6.2: Arquitetura do Controlador Neural

$$tansig(n) \quad = \frac{2}{(1+exp(-2*n))-1} \qquad purelin(n) = n \tag{6.18}$$

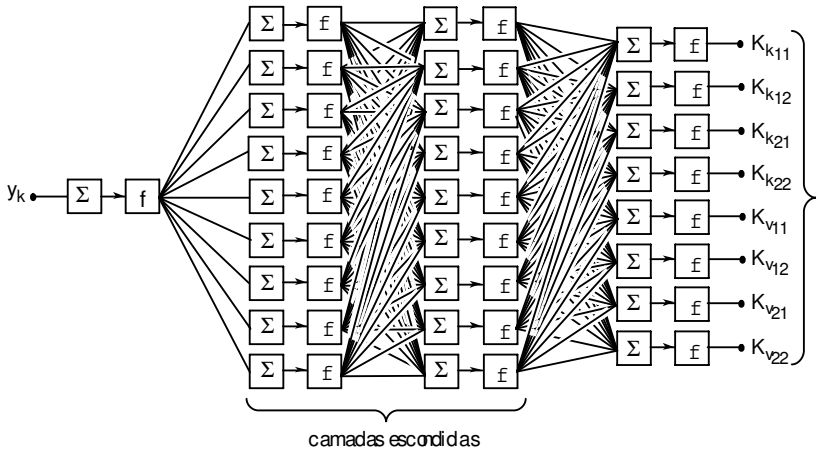

Fig. 6.3: Rede Neural Feedforward

Algoritmo de Aprendizado

Para um conjunto de dados de entrada-saída para treinamento, o conjunto de condições que descreve o processo de aprendizado para o *Perceptron* multicamada é o algoritmo *Backpropagation*,[138].

O algoritmo *backpropagation* é um algoritmo supervisionado que utiliza pares (entrada, saída desejada) para, por meio de um mecanismo de correção de erros, ajustar os pesos da rede. O treinamento ocorre em duas fases, em que cada fase percorre a rede em um sentido. Estas duas fases são chamadas *forward* e *backward*. A fase forward é utilizada para definir a saída da rede para um dado padrão de entrada. A fase backward utiliza a saída desejada e a saída fornecida pela rede para atualizar os pesos de suas conexões.

Quando criamos uma *Rede Neural* temos que ajustar os pesos em cada um dos neurônios que estão na rede. Para isto existem fórmulas que dependem de se o neurônio é de saída ou se é um neuronio intermédiario. Para compreender melhor o raciocíonio para construir o algoritmo, consideremos uma RNA com L camadas, das quais $L-1$ camadas são ocultas; a saída do neurônio i vai ser igual à:

$$x_{i,p}^{l} = \phi(\xi_{i,p}^{l}) \quad i = 1, \ldots, l \qquad \xi_{i,p}^{l} = \sum_{j}^{n} w_{i,j}^{l} x_{j,p}^{l-1} \qquad l = 1, \ldots, L \tag{6.19}$$

onde l é um índice que identifica as diferentes camadas, p constitui um índice padrão, $\phi(.)$ é a função de ativação nas diferentes camadas da rede; a constante n representa o número de conexões de entrada do nodo i, o neurônio i contém m entradas e w_{ij}^{l} são os pesos sinápticos da conexão entre a entrada $x_{j,p}^{l}$ e o nodo i na entrada j da camada l. São conhecidos P padrões de entrada e suas correspondentes saídas desejadas. Nosso objetivo é minimizar o erro total E de todos os neurônios na camada de saída.

Representamos tal problema pela minimização da seguinte função de custo:

$$min_{w_{i,j}^l} E = \frac{1}{p}\sum_{p=1}^{P} E_p \qquad E_p = \frac{1}{2}\sum_{i=1}^{N_l}(x_{i,p}^d - x_{i,p}^l)^2 \tag{6.20}$$

onde $x_{i,p}^l$ é a i-ésima saída em relação em relação ao p-ésimo padrão de entrada na camada l gerada pela rede, $x_{i,p}^d$ corresponde à i-ésima saída desejada em relação ao p-ésimo padrão de entrada, E_p é a medida do padrão p-ésimo na função de custo E e finalmente N_l define o número de neurônios na camada l.

O algoritmo backpropagation é baseado na Regra Delta, sendo também chamada de Regra Delta Generalizada; ele propõe uma forma de definir o erro dos nodos das camadas intermediárias, possibilitando o ajuste de seus pesos, utilizando-se o método do gradiente.

A função de custo a ser minimizada é uma função de erro ou energia, definida pela equação (6.20). Esta equaçao define o erro total cometido pela rede, ou a quantidade em que para todos os padrões p de um dado conjunto, as saídas geradas pela rede diferem das saídas desejadas. A seguir apresentamos o cálculo da Regra Delta.

$$\begin{cases} \Delta w_{i,j}^l & = -\eta\frac{\partial E(n)}{\partial w_{i,j}^l} \\ & = \eta\delta_{i,p}^l x_{j,p}^{l-1} \\ \delta_{i,p}^l & = -\eta\frac{\partial E(n)}{\partial \xi_{i,p}^l} \\ & = (x_{i,p}^d - x_{i,p}^L)\phi'(\xi_{i,p}^L) \end{cases} \tag{6.21}$$

onde η é a taxa de aprendizagem da rede neural.

Treinamento

O esquema de treinamento do escalonador de ganhos neural é apresentado na Fig. 6.4. Neste esquema, o treinamento é feito na forma usual, baseado no conjunto de N pontos de operação $r_\sigma, \sigma \in I$ escolhidos, os quais constituem as N colunas da matriz de referência $W = \begin{bmatrix} r_1 & \dots & r_N \end{bmatrix}$ e nos correspondentes N controladores ótimos que constituem as N linhas da matriz de controladores ótimos K, chamados alvos, dados como entrada.

Para criar a Rede Neural Perceptron usamos as ferramentas apresentadas no Matlab; neste caso criamos uma Rede Neural *backpropagation Feed-Forward* com 4 camadas:

```
net=newff(V1,[2 9 9 8],{'tansig''purelin''purelin' 'purelin'});
```

O primeiro parâmetro para a geração da RNA, definida como *net*, é uma matriz, $V_1 \in R^{m\times 2}$, onde m corresponde ao número de entradas do sistema e o segundo parâmetro é fixo pois corresponde aos valores mínimo e máximo da referência, $\breve{r}_k$.

A seguir é feito o treinamento (*train*) da rede com a matriz de referências W e os alvos correspondentes K.

```
[net,Y,E] = train(net,W,K);
```

O treinamento ocorre até que o número máximo de épocas aconteça ou até que o objetivo de desempenho seja alcançado.

Posteriormente uma simulação (*sim*) é realizada, obtendo-se uma matriz de ganhos neurais G:

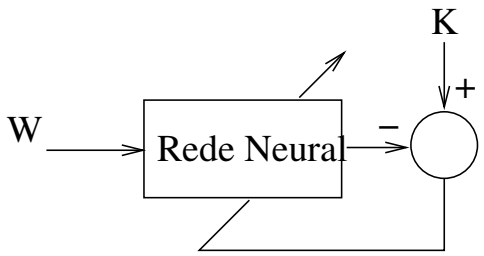

Fig. 6.4: Esquema de Treinamento do Controlador Inteligente LPV

```
G = sim(net,W)
```

Com os resultados obtidos fazemos uma verificação para constatar se o erro obtido é mínimo, ou seja, se esta rede neural obtém matriz de ganhos neurais semelhantes às oferecidas na matriz K, correspondente aos controladores ótimos obtidos para cada modelo invariante no tempo identificado, nos respectivos pontos de operação analisados.

Projeto do Controlador LPV Inteligente

Para o projeto do controlador neural com escalonamento de ganhos, *Controlador Inteligente*, escolhemos um modelo identificado, dentre os nove modelos obtidos e definimos uma nova matriz de referência $\bar{W}_{m\times p} = \left[\begin{array}{ccc} r_1 & \ldots & r_p \end{array} \right]$, para a qual projetamos a lei de controle seguidor ótima que além do desempenho desejado assegure a estabilidade do sistema. Com esta nova matriz de referência estamos interessados em fazer com que a cada N/p pontos o sistema acompanhe cada um dos p valores de referência da matriz $\bar{W}$ a partir da primeira referência, ou seja, primeiramente o valor de referência r_1, depois de N/p pontos, a referência r_2 e assim por diante.

Calcula-se o valor de $\breve{v}_{\bar{W}}$, segundo (6.13), como sendo $\breve{v}_{\bar{W}} = C_m^T L\bar{W}$; determina-se para o k-ésimo instante de tempo, uma matriz A_k variante no tempo. Faz-se um ajuste on-line dos parâmetros do controlador inteligente para todo o conjunto I, fazendo simulações neurais com a rede *net* gerada na identificação, com a nova matriz de referência $\bar{W}$:

$$\bar{G} = sim(net, \bar{W}) \tag{6.22}$$

Desta forma, pela capacidade de generalização do controlador neural com EG, materializa-se o ajuste da saída para qualquer referência oferecida dentro da faixa de operação de interesse.

Uma abordagem alternativa incluíndo incertezas aditivas aleatórias na dinâmica dos modelos identificados é proposta, resultando num controlador suficientemente robusto para estabilizar a planta variante no tempo:

$$\breve{v}_k \;\; = [[A_m + (diag(eig(A)) - A_m) * 0.01 * randn] - B_m\breve{K}_k]^T\breve{v}_{k+1} + C_m^T Q\bar{W} \tag{6.23}$$

onde A_m, B_m, C_m denotam as matrizes identificadas transformadas.

6.3 Experimentação e Resultados

Através dos algoritmos numéricos, para um número finito N de pontos de operação adaptivamente controlados, diferentes conjuntos de quadrúplas de matrizes do sistema são identificados. Assim o algoritmo $MOESP\_VAR$, [151], é executado N vezes para determinar os conjuntos de quadrúplas de matrizes para estes N pontos de operação. Para cada um destes pontos de operação, um controlador linear seguidor ótimo é projetado. Fazemos alguns experimentos para determinar estes controladores lineares. Generalizamos todos estes resultados para qualquer ponto de referência na região de operação do sistema.

É utilizada uma Rede Neural MLP para a implementação dos controladores *gain scheduling* para plantas lineares MIMO. São selecionados diversos pontos de operação de modo a cobrir toda a faixa de operação da planta. Estes pontos de operação são determinados pelos valores da referência. Para um número finito N de pontos de operação modela-se computacionalmente no espaço de estado a planta variante no tempo sendo obtidas matrizes $A(\sigma), B(\sigma), C(\sigma)$, $\sigma \in N$ que são aproximações invariantes no tempo para um conjunto finito de instantes de tempo. São então calculados os valores das matrizes de ganhos $\breve{K}_k, \breve{K}_k^v$ para então finalmente, a partir destes valores, projetar os controladores ótimos $\breve{u}_k$ correspondentes.

Planta Variante no Tempo

Consideremos o sistema discreto linear variante no tempo

$$\begin{cases} x_{k+1} = & A_k x_k + B_k u_k \\ y_k = & C_k x_k \end{cases} \tag{6.24}$$

A_k esta definida segundo a expressão (6.25) e é uma matriz estável para todo k:

$$A_k = \begin{bmatrix} a1 & 0 \\ 0 & a2_k \end{bmatrix} \tag{6.25}$$

onde

$$\begin{aligned} a1 &= -0.3 \\ a2_k &= -1/3 - 0.1 sen(2\pi k/400) \end{aligned} \tag{6.26}$$

Para esta planta variante no tempo tomamos $N = 150$ pontos diferentes, $k = 1, 2, \ldots, 150$. Os autovalores (Av) respectivos de A_k, estão variando com o tempo na seguinte forma:

$$\begin{aligned} Av(1) = \begin{bmatrix} -0.3349 \\ -0.3000 \end{bmatrix}; \quad Av(2) = \begin{bmatrix} -0.3365 \\ -0.3000 \end{bmatrix}; \quad Av(3) = \begin{bmatrix} -0.3380 \\ -0.3000 \end{bmatrix}; \\ \vdots \\ Av(148) = \begin{bmatrix} -0.4062 \\ -0.3000 \end{bmatrix}; \quad Av(149) = \begin{bmatrix} -0.4051 \\ -0.3000 \end{bmatrix}; \quad Av(150) = \begin{bmatrix} -0.4040 \\ -0.3000 \end{bmatrix} \end{aligned} \tag{6.27}$$

Para estas 150 matrizes A_k, são calculados um vetor de estado x_k e um vetor de saída y_k, para um valor de entrada u_k gerado aleatoriamente. Na Fig.6.5 apresentamos a trajetória dos elementos de $a2_k$ pertencentes a matriz variante no tempo A_k e as saídas respectivas y_k, as quais são apresentadas no Anexo 8.4.

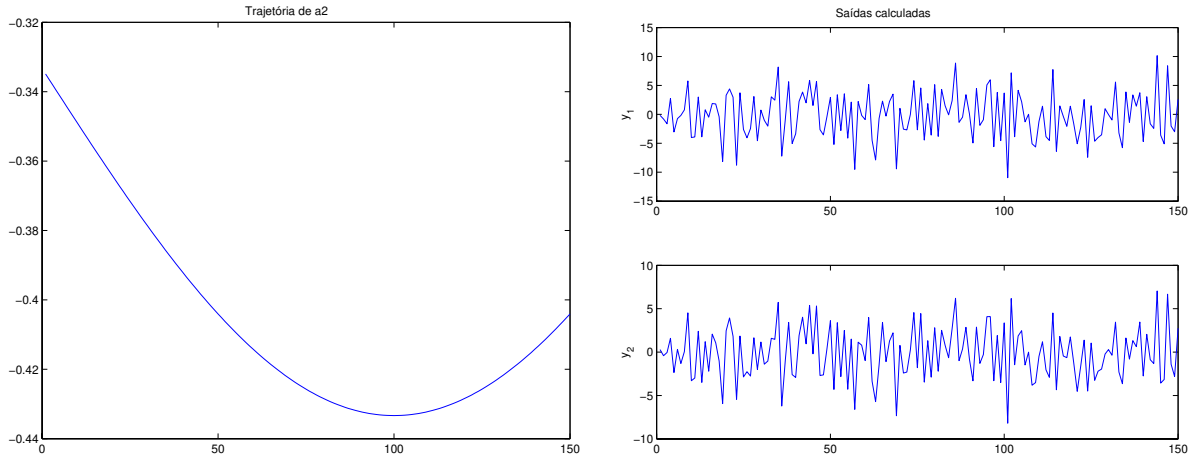

Fig. 6.5: Trajetória dos elementos de $a2_k$ e das saídas y_k, respectivamente

Para tal conjunto de vetores de entrada-saída, fazemos a identificação da planta variante no tempo. As matrizes obtidas na identificação, $(A_c(\sigma), B_c(\sigma), C_c(\sigma), D_c(\sigma))$:

$$A_c = \begin{bmatrix} -0.3879 & 0.4837 \\ 0.0041 & -0.3224 \end{bmatrix} \quad B_c = \begin{bmatrix} -0.9943 & -5.3566 \\ 0.6871 & -0.0689 \end{bmatrix}$$
$$C_c = \begin{bmatrix} -0.7530 & 0.3808 \\ -0.5501 & -0.7859 \end{bmatrix} \quad D_c = \begin{bmatrix} -0.0232 & -0.0047 \\ -0.0151 & -0.0029 \end{bmatrix} \tag{6.28}$$

sofrem alguma variação de acordo com a mudança dos autovalores da matriz A_k. Para garantir satisfazer o sistema original variante no tempo, criamos matrizes identificadas modificadas, $(A_m(\sigma), B_m(\sigma), C_m(\sigma), D_m(\sigma))$, com a seguinte estrutura:

```
[Av,V]=eig(Ac);
Am = inv(Av)*Ac*Av;
Bm = inv(Av)*Bc;
Cm = Cc*Av ;
Dm=Dc
```

$$Av = \begin{bmatrix} -0.9989 & -0.9839 \\ 0.0462 & -0.1787 \end{bmatrix}; \quad V = \begin{bmatrix} -0.4102 & 0 \\ 0 & -0.3000 \end{bmatrix} \tag{6.29}$$

e as matrizes identificadas modificadas são:

$$Am = \begin{bmatrix} -0.4102 & 0 \\ 0 & -0.3000 \end{bmatrix}; \quad Bm = \begin{bmatrix} 3.8107 & 3.9707 \\ -2.8583 & 1.4129 \end{bmatrix}$$
$$Cm = \begin{bmatrix} 0.7698 & 0.6728 \\ 0.5132 & 0.6817 \end{bmatrix}; \quad Dm = \begin{bmatrix} -0.0232 & -0.0047 \\ -0.0151 & -0.0029 \end{bmatrix} \tag{6.30}$$

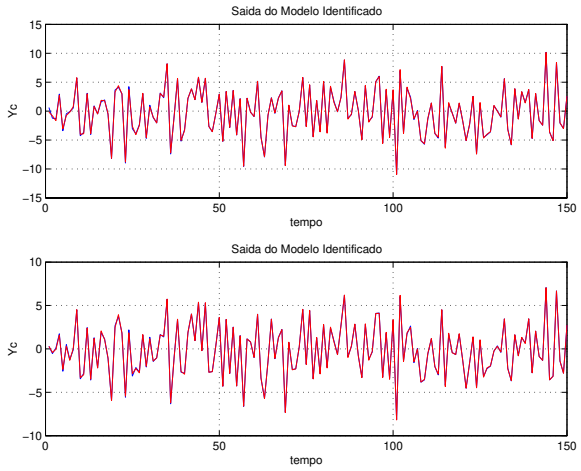

Fig. 6.6: Trajetórias dos elementos na matriz de saída Y_c

Na Fig.6.6 apresentamos as curvas correspondentes as duas componentes das saídas Y_c obtidas apos a transformação do modelo identificado. Pode-se observar que estas saídas coincidem em todo o intervalo de tempo calculado com as saídas y_k do *Benchmark* (6.24).

Para uma referência r_σ:

$$r_\sigma = \begin{bmatrix} 5 \\ 1 \end{bmatrix} \tag{6.31}$$

fazemos os cálculos das matrizes de ganho do controlador, $\breve{K}_k, \breve{K}_k^v$, da solução $\breve{P}_k$ da equação recursiva de Riccati discreta e do vetor $\breve{v}_k$ relacionado à referência r_σ, segundo as equações expressas em (6.16) e (6.17). Os resultados obtidos são:

$$\breve{K}_k = \begin{bmatrix} -0.0348 & 0.0695 \\ -0.0694 & -0.0680 \end{bmatrix} \quad \breve{K}_k^v = \begin{bmatrix} 0.1203 & -0.1128 \\ -0.0142 & 0.0162 \end{bmatrix} \quad \breve{P}_k = \begin{bmatrix} 59.2595 & 62.5628 \\ 62.5642 & 68.9887 \end{bmatrix} \tag{6.32}$$

Calcula-se posteriormente o vetor de controle $\breve{u}_k \in (2 \times N)$ para a referência r_σ definida acima.

A seguir na Fig.6.7 apresentamos as duas componentes da saída yc_k do sistema controlado e na Fig.6.8 apresentamos as duas componentes do sinal de controle do sistema identificado.

Todo este processo é repetido para várias referências $\breve{r}_k$. Para o exemplo que estamos apresentando, o mesmo foi repetido nove vezes. Os respectivos nove valores para $\breve{r}_k$ são apresentados, por colunas, na matriz W,

$$W = \begin{bmatrix} 1 & 1 & 1 & 3 & 3 & 3 & 5 & 5 & 5 \\ 1 & 3 & 5 & 1 & 3 & 5 & 1 & 3 & 5 \end{bmatrix} \tag{6.33}$$

Para cada um destes pontos de operação, colunas da matriz W , obtém-se um modelo identificado no espaço de estado, cujas matrizes $A_m(\sigma), B_m(\sigma), C_m(\sigma)$ satisfazem o sistema original variante no tempo. Para cada um destes pontos de operação, são calculadas as matrizes de ganho $\breve{K}_k, \breve{K}_k^v$. Por exemplo, para três (3) pontos de operação, obtém-se:

- Ponto de Operação 1: Ganhos obtidos:

$$\breve{K}_k = \begin{bmatrix} -0.0351 & 0.0655 \\ -0.0690 & -0.0683 \end{bmatrix} \quad \breve{K}_k^v = \begin{bmatrix} 0.1242 & -0.1166 \\ -0.0155 & 0.0175 \end{bmatrix} \tag{6.34}$$

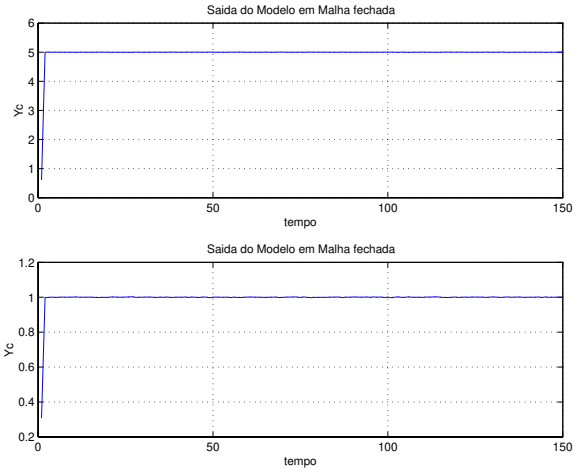

Fig. 6.7: Saída do Sistema Controlado Y_c

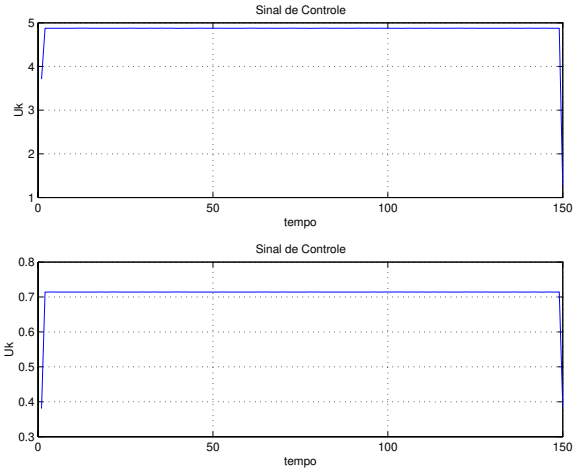

Fig. 6.8: Sinal de controle

- Ponto de Operação 5: Ganhos obtidos:

$$\breve{K}_k = \begin{bmatrix} -0.0351 & 0.0628 \\ -0.0695 & -0.0671 \end{bmatrix} \quad \breve{K}_k^v = \begin{bmatrix} 0.1291 & -0.1212 \\ -0.0152 & 0.0172 \end{bmatrix} \tag{6.35}$$

- Ponto de Operação 9: Ganhos obtidos:

$$\breve{K}_k = \begin{bmatrix} -0.0352 & 0.0686 \\ -0.0706 & -0.0678 \end{bmatrix} \quad \breve{K}_k^v = \begin{bmatrix} 0.1215 & -0.1136 \\ -0.0141 & 0.0161 \end{bmatrix} \tag{6.36}$$

Finalmente, para cada um destes pontos de operação, projetamos o controlador linear seguidor ótimo $\breve{u}_k$ correspondente.

Neste exemplo, as faixas de operação da matriz de referência, (6.33) vão de 1 até 5, por isto a matriz V_1, utilizada para criar a RNA é definida como:

$$V_1 = \begin{bmatrix} 1 & 5 \\ 1 & 5 \end{bmatrix} \tag{6.37}$$

A RNA feedforward é treinada para 100 épocas de tempo com a matriz de referência W e uma matriz K formada pelos controladores ótimos, aqui chamado *alvos*, dados como entradas; estes correspondem às matrizes de ganho obtidas $\breve{K}_k, \breve{K}_k^v$ para cada ponto de operação analisado.

A Fig.6.9 apresenta o desempenho do treinamento da RNA, utilizando o algoritmo backpropagation, em relação ao mapeamento dos pontos de operação com os controladores ótimos. O erro quadrático médio foi reduzido de 7.98622 até $5.67857e-007$, em aproximadamente 10 épocas.

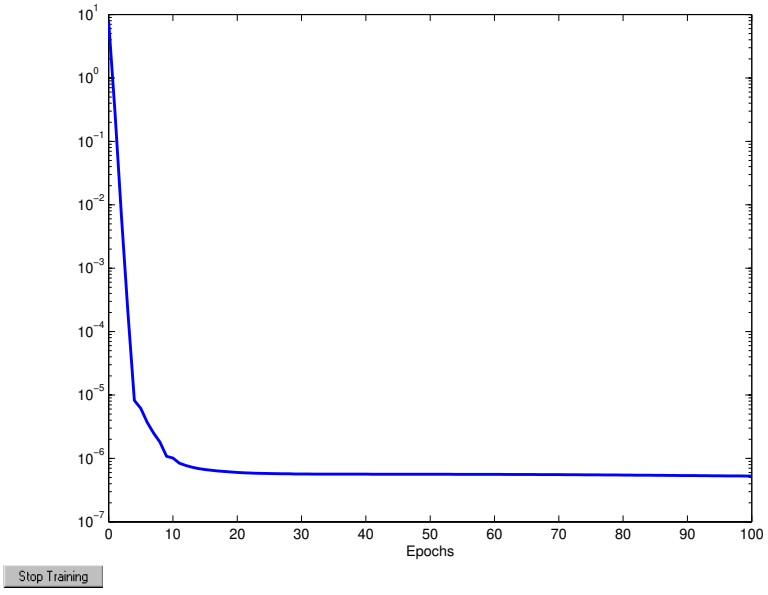

Fig. 6.9: Treinamento Neural por Backpropagation

Para o projeto do controlador neural com escalonamento de ganhos, escolhemos um modelo identificado, dentre os nove modelos obtidos:

$$Am = \begin{bmatrix} -0.4010 & 0 \\ 0 & -0.3000 \end{bmatrix} \quad Bm = \begin{bmatrix} 3.9588 & 3.7956 \\ -2.9939 & 1.5231 \end{bmatrix} \quad Cm = \begin{bmatrix} 0.7723 & 0.6795 \\ 0.5148 & 0.6750 \end{bmatrix} \tag{6.38}$$

e definimos uma nova matriz de referência $\bar{W}$,

$$\bar{W} = \begin{bmatrix} 5 & 20 & 35 & 25 & 15 & 1 \\ 5 & 20 & 35 & 25 & 15 & 1 \end{bmatrix} \tag{6.39}$$

para a qual projetamos a lei de controle seguidor ótima. Com esta nova matriz de referência estamos interessados em fazer com que a cada $N/6$ pontos o sistema acompanhe cada um dos valores das referências desta matriz a partir da primeira, ou seja, primeiramente o valor de referência $\begin{bmatrix} 5 \\ 5 \end{bmatrix}$, depois de $N/6$ pontos, a referência $\begin{bmatrix} 20 \\ 20 \end{bmatrix}$ e assim por diante.

A Fig.6.10 apresenta o desempenho da saída do sistema variante no tempo controlado inteligentemente por um controlador seguidor ótimo neural com escalonamento de ganhos a partir da matriz de referência apresentadas em (6.39), em presença de ruído na saída (média=0 e variância = 0.1) e com mecanismo de ajuste on-line dos parâmetros $\breve{K}_k$ and $\breve{K}_k^v$ do controlador. A Fig.6.11 mostra o sinal de controle inteligente, para a mesma matriz de referência.As Figs.6.10 e 6.11 apresentam as duas componentes respectivamente da saída Y_c e do controle U_k.

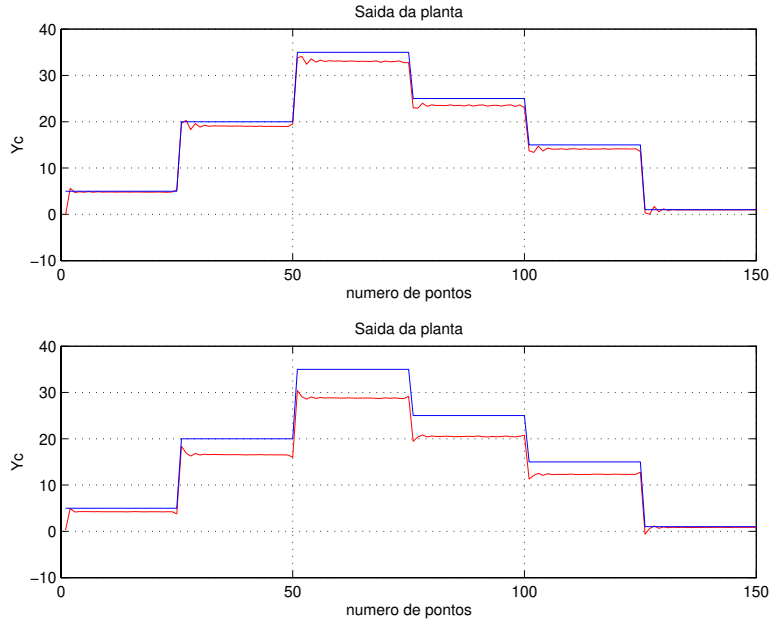

Fig. 6.10: Saída do Sistema Controlado Inteligentemente

Na Fig.6.12 são mostradas superfícies de ganhos neurais para o controlador ILPV projetado e implementado para um conjunto de pontos de referência deste sistema MIMO discreto variante no tempo. Para cada ponto de operação é mostrada a saída da RNA correspondente para uma grade de referências com dimensões $[40, 0] \times [0, 40]$. Para cada elemento de cada matriz de ganho K_k e K_K^v é apresentada uma superfície de ganho neural para um conjunto de pontos de referência escolhido, por exemplo (6.39). Pode-se observar que a transição entre os ganhos do controlador inteligente para os diferentes pontos de operação se faz de forma suave ressaltando a propriedade de generalização que possui uma RNA. Esta de fato fica ainda mais ressaltada quando se considera a matriz W usada no

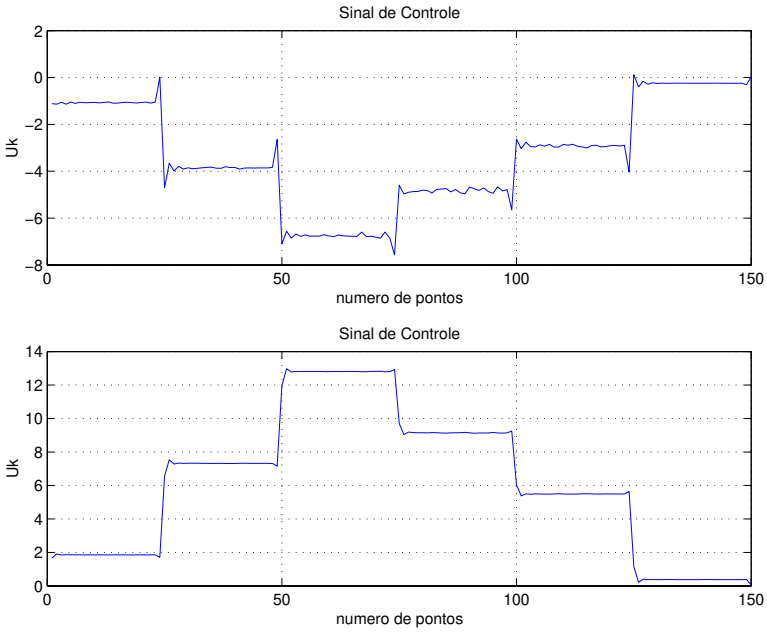

Fig. 6.11: Sinal de Controle Inteligente

treinamento e os resultados apresentados nas Figs. 6.10 - 6.12. Portanto, destes resultados podemos concluir que esta estrutura de controle inteligente fornece uma suave interpolação dos parâmetros do controlador para seus diversos pontos de operação e é capaz de generalizar para uma ampla faixa de operação, o treinamento realizado em faixa relativamente mais estreita.

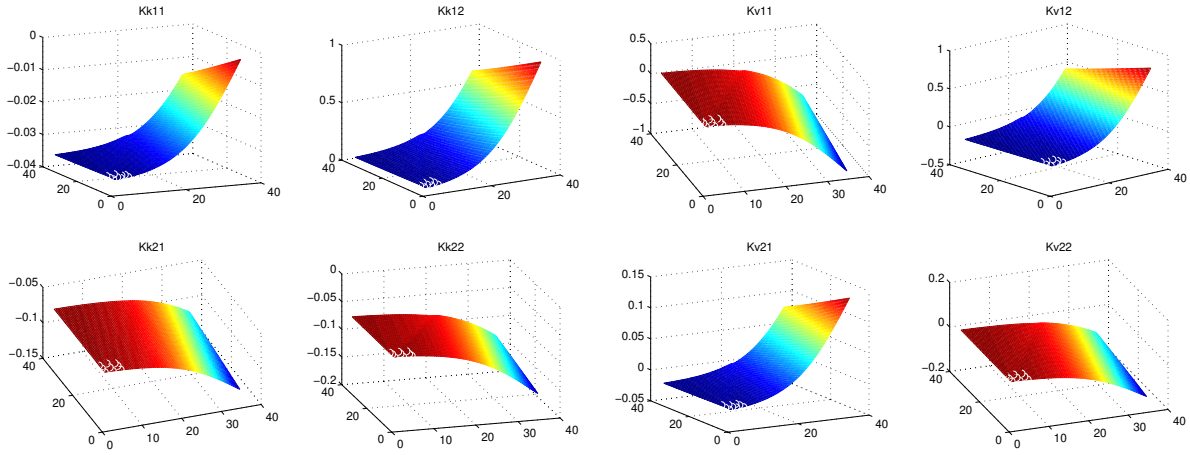

Fig. 6.12: Superficies de ganhos neurais para um conjunto de referências

Capítulo 7

Conclusões

Neste trabalho de tese, o problema de **Modelagem Computacional de Dados** de Sistemas e de Séries Temporais **Multivariáveis no espaço de estado** por métodos de subespaço é considerado, assim como o **Controle Inteligente** no espaço de estado, objetivando modelar e controlar sistemas dinâmicos lineares discretos no tempo no espaço de estado.

No Capítulo 3 formulamos e implementamos um procedimento computacional para identificação de sistemas lineares multivariáveis não-estacionários discretos no espaço de estado, sob a suposição de que as mudanças da dinâmica do sistema ocorram lentamente, fundamentado em um método eficiente de identificação para sistemas estacionários do tipo **MOESP**, e o chamamos Algoritmo **MOESP_VAR**.

Pelos resultados obtidos e apresentados neste trabalho, podemos dizer que nossa proposta numérica para sistemas variantes no tempo cujas matrizes variam lentamente, mediante a utilização de métodos de subespaço para sistemas invariantes, é viável.

Com base nos estudos expostos nos Capítulo 2 e 3, elaboramos uma proposta que chamamos Algoritmo **MOESP_AOKI** para Modelagem Combinada Determinística-Estocástica de dados no espaço de estado utilizando a experiência adquirida e algoritmos tratados neste trabalho e em [12, 32]. Apresentamos também alguns resultados obtidos com uma versão do método de subespaço **Numerical algorithms for Subspace State System IDentification (N4SID)** que faz identificação Determinística-Estocástica no espaço de estado, para comparação com nossa proposta, para a qual também apresentamos os primeiros resultados computacionais.

Como conclusão podemos dizer que nossa proposta de algoritmo **MOESP_AOKI** obtém resultados melhores que o algoritmo **N4SID** quando este é aplicado numa forma combinada. Por outro lado, independentemente desta comparação, concluímos que o algoritmo **MOESP_AOKI** proposto tem um excelente desempenho em identificação de sistema sujeito a ruído, tanto para ruído branco como para ruído colorido.

Os algoritmos para solução neural da *EARD*, da *EARC*, da *IARD* e da *IARC*, propostos, implementados e ilustrados nos exemplos de aplicação a problemas de controle ótimo, respectivamente, tendo em mente aplicações em tempo real, mostram-se viáveis, eficientes e úteis, além de confirmarem a exatidão de cada solução obtida por métodos diversos, em especial pelos que propusemos em [149, 150] para a *EARD* e para a *IARD*.

A arquitetura de controle inteligente para criar esquema de controle ILPV, proposta e implementada nesta tese de doutorado, para controle de plantas lineares discretas multivariáveis variantes no

tempo foi bem sucedida em adição a ser original em sua concepção e estrutura. Estudos adicionais, análises e aplicações são desenvolvimentos naturais e necessários como consequência deste estudo.

Através de exemplos, os resultados de simulações mostram que o controlador neural proposto pode representar a dinâmica global dos controladores lineares projetados para diversos pontos de operação da planta.

Dos resultados apresentados para este sistema de controle multivariável inteligente podemos afirmar que:

- ele melhora o desempenho da malha por causa da otimização;
- ele funciona como um adaptador inteligente atualizando todos os oito (8) parâmetros ajustadores e provendo interpolação entre condições de operação devido ao escalonador neural de ganhos.
- ele segue trajetórias a despeito de ruído e é eficiente no controle de plantas lineares variantes no tempo.

Propomos como trabalhos futuros:

- Implementar computacionalmente o algoritmo *AVW* para sistemas variantes no tempo: algoritmo MOESP_AOKI_VAR, que aqui também propomos.
- Realizar desenvolvimentos teóricos e computacionais de propostas para solução utilizando Redes Neurais da Equação Recorrente de Riccati Discreta apresentada no Capítulo 6, para a solução do problema de controle seguidor ótimo e da Equação Algébrica de Riccati Discreta apresentada no algoritmo **MOESP_AOKI**.
- Fazer uma análise da métrica do algoritmo $MOESP\_VAR$, ou seja, analisar se verdadeiramente as variações das janelas acompanham os resultados da identificação e descrevem adequadamente os dados a modelar.
- Realizar Modelagem de Séries Temporais Variantes no tempo no espaço de estado através de algoritmo $AOKI\_VAR$.
- Realizar estudos adicionais e aplicações relativas ao controlador *ILPV* proposto no Capítulo 6.
- Realizar controlador *ILPV* com aumento do modelo da planta por inclusão de integrador para atenuar ou eliminar erros de regime permanente.
- Realizar o Controle Inteligente em contexto de Controle Adaptativo Estocástico Multivariável, utilizando na identificação de dados ruidosos o algoritmo *MOESP_AOKI_VAR* e a estrutura de controle inteligente LPV proposta no Capítulo 6.

Referências Bibliográficas

[1] AI-Akhras M.A., Aly G.M. and Green R.J., "Neural Network Learning Approach of Intelligent Multimodel Controller", *IEE* Proc.-Control Theory Appl., Vol. 143, No.4, July, 1996.

[2] Akaike, Hirotugu, "Stochastic Theory of Minimal Realization", IEEE Transaction on Automatic Control, Vol.AC-19, Nr.6, 667-674, 1974.

[3] Apkarian, P. and Gahinet P. and Becker G., "Self-scheduled H_∞ Control of Linear Parameter-Varying Systems: a design example", Automatica, Vol.31, pp.1251-1261, 1995.

[4] Aoki, Masanao, "State Space Modeling of Time Series", Springer-Verlag, 1987.

[5] Aoki, Masanao and Havenner, Arthur, "State Space Modeling of Multiple Time Series", Econometric Reviews, Vol.10, Nr.1, 1-59, 1991.

[6] Arnold, W.F. and Laub, Alan J., "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations, Proceedings of the *IEEE*, 72 (12), 1984.

[7] Aström, Karl J. and Eykhoff, P., "System Identification - A Survey", Automatica, Volume 7, N_0 2, 123-162, March, 1971.

[8] Aström, Karl J. and Wittenmark, Björn, "Adaptive Control", 2^{nd} edition, Addison-Wesley, 1995.

[9] Back, T. and U.Hammel and H.P.Schwefel, "Evolutionary Computation: Comments on the History and Current State", IEEE Transaction on Evolutionary Computation, Vol.1, No.1, 1997.

[10] Balakrishnan, V. and Feron, E., "Linear Matrix Inequalities in Control Theory and Applications", Int. J. Robust Nonlinear Contr., Vol.6, pp.869-1099, 1996.

[11] Barbosa, K.A., Trofino, A., "Técnicas LMI para Análise de Sistemas com Restrições Algébricas no Estado", Revista Controle & Automação, Vol.13, N_0 1, Jan-Abril 2002.

[12] Barreto, G., "Modelagem Computacional Distribuída e Paralela de Sistemas e de Séries Temporais Multivariáveis no Espaço de Estado", Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, UNICAMP, 2002.

[13] Becker, G. and Packard, A., "Robust Performance of Linear Parametrically-Varying Systems using Parametrically-dependent Linear Feedback", Sys. Contr. Lett., Vol.23, pp.436-442, 1994.

[14] Bendtsen, J. and Trangbaek, K., "Transformation of Neural State Space Models into LFT Models for Robust Control Design", In Proc. of the International Conference of Control, Automation, Robotics and Vision, 2000.

[15] Bendtsen, J. and Trangbaek, K., "Robust Quasi-LPV Control Based on Neural State-Space Models", IEEE Transactions on Neural Networks, Vol.13,$N_0 2$, March 2002.

[16] Bendtsen, J. and Trangbaek, K., "Discrete-Time LPV Current Control of an Induction Motor", in Proc. of the 42nd IEEE Conference on Decision and Control 2003.

[17] Bertsekas, D. and Tsitsiklis, J., Parallel and Distributed Computations, Prentice Hall, 1989.

[18] Billings, S.A. and S. Chen, "Neural Networks and System Identification", Neural Networks for Control and Systems, pp.181-205, In Warwick K.,Irwin G.W. and Hunt K.J. editors, 1992.

[19] Billings, S.A. and Zhu, Q.M., "Nonlinear Model Validation Using Correlation Tests", Int. J. Control 60, 1107-1120, 1994.

[20] Bottura, C.P., Barreto, G., Bordon, M.J. e Costa Filho, J.T., "State Space Modelling and High Performance Computing of Multivariate Time Series", Eigth SIAM Conference on Parallel Processing for Scientific Computing, March, 1997.

[21] Bottura, C.P., Teixeira, M.M. and Bordon, M.J., "Aplicação de um Controlador Neural-Deslizante na Estabilização de um Pêndulo Invertido com o Cerebellar Model Articulation Controller", V Simpósio Brasileiro de Redes Neurais, 9-11 Dezembro, 1998.

[22] Bottura, C.P., Tamariz, ADR., Fonseca, JV. and Barreto, G., "Sequential and Parallel Algebraic Riccati Equations Solutions via ESST on the Schur Method", Proceedings of the 38^{th} IEEE Conference on Decision and Control, CDC, pp.2739-2740, 1999.

[23] Bottura, C.P., Barreto, G.,Bordon, M.J. and Tamariz, A.D.R., "Tratamento Computacional de Alto Desempenho em Método de Subespaços para Modelagem de Dados", XV Congresso Brasileiro de Engenharia Mecânica - Cobem, Águas de Lindóia, São Paulo, Brasil, 1999.

[24] Bottura, C.P., Barreto, G.,Bordon, M.J. and Tamariz, A.D.R., "Parallel and Distributed State - Space Modeling for Computation of Time Series using Realization theory, *Proceedings of the Third International Symposium on Mathematical Modelling*, Vienna, Austria, February 2-4, 2000.

[25] Bottura, C.P., Barreto, G.,Bordon, M.J. and Tamariz, A.D.R., "Parallel and Distributed Computational Data Modelling via Verhaegen $\&$ Dewilde's subspace method". *Proceedings of the 2000 American Control Conference*, (ACC'2000), June 28-30, Chicago, Illinois, USA, 2000.

[26] Bottura, C.P., Barreto, G., Bordon, M.J. and Tamariz, ADR.,"An Approach to State Space Computational Modeling and Prediction of Time Series in Parallel and Distributed Computers". XVI edição do Congresso Brasileiro de Engenharia Mecânica (COBEM), novembro, 2001.

[27] Bottura, C.P., Barreto, G., Tamariz, A.D.R. and M.J. Bordon, "Parallel and Distributed Computational Multivariate Time Series Modeling in the State Space", *Proceedings of the American Control Conference*, (ACC'2002), Anchorage, Alaska, USA, pg 1466-1471, Maio 2002.

[28] Bottura, C.P., Barreto, G., Tamariz, A.D.R., Cáceres, A.F.T., "Parallel and Distributed MOESP Computational System's Modelling", *Proceedings of the* 10^{th} *Mediterranean Conference on Control and Automation*, (MED'2002), Lisboa Portugal, July 9-12, 2002.

[29] Box, George E.P. and Jenkins, Gwilym M., "Time Series Analysis: Forecasting and Control", Holden-Days Series in Time Series Analysis and Digital Processing, Holden-Day, 1976.

[30] Bunse-Gerstner, Angelika and Faβbender, Heike, "A Jacobi-Like Method for Solving Algebraic Riccati Equations on Parallel Computers", IEEE Transactions on Automatic Control, Vol.42, No.8, pp.1071–1084, August, 1997.

[31] Bryson, A.E. and Ho, Y.C., "Applied Optimal Control", New York: Hemisphere, 1975.

[32] Cáceres, Angel Fernando Torrico, "Identificação e Controle Estocásticos Decentralizados de Sistemas Interconectados Multivariáveis no Espaço de Estado", Tese de Doutorado a ser apresentada na FEEC da UNICAMP, em 2005.

[33] Caines, P.E., "Linear Stochastic Systems", Wiley, 1988.

[34] Chan, W.S. and Tong, H., "On Tests for Nonlinearity in Time Series", J. of Forecasting 5, 217-228, 1996.

[35] Chen, Lingji and Narendra, K.S., "Nonlinear Adaptive Control using Neural Networks and Multiple Models", Automatica 37, 1245-1255, 2001.

[36] Chen, F.C. and Khalil, H.K., "Adaptive Control of Nonlinear Discrete-Time Systems Using Neural Networks", IEEE Transaction on Automatic Control, Vol.40, No.5, pp. 791-801, May, 1995.

[37] Chou, J.H., Hsieh, C.H., Sun, J.H., "On-Line Optimal Tracking Control of Continuous-Time Systems", Mechatronics 2004.

[38] Chun-Liang Lin and Chi-Chih Lai and Teng-Hsien Huang, "A Neural Network for Linear Matrix Inequality Problems", IEEE Transactions on Neural Networks, Vol.11, No.5, September, 2000.

[39] Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function", Mathematics of Control, Signals and Systems, Vol.2, No.4, pp.303-314, 1989.

[40] De Moor, Bart and Van Overschee Peter, "Numerical Algorithms for State Space Subspace Systems Identification", Editado por alberto Isidori em Trends in Control - A European Perspective, 385-422, springer-Verlag, 1995.

[41] Dewilde, Patrick and Van der Veen, Alle-Jan, "Time-Varying Systems and Computations", Kluwer Academic Publishers, 1998.

[42] Eykhoff, Pieter, "System Identification - Parameter and State Estimation", John Wiley & Sons, 1974.

[43] Etxebarria, V., "Adaptive Control of Discrete Systems Using Neural Networks", IEE Proc. Control Theory Appl., Vol. 141,N_o. 4, July 1994.

[44] Faurre P.L., "Stochastic Realization Algorithms in System Identification: Advances and Case Studies", ed. Mehra, Raman K. and Lainiotis, Dimitri G., Academic Press, 1976.

[45] Favoreel, Wouter, "Subspace Methods for Identification and Control of Linear and Bilinear Systems", Tese de Doutorado, 1999.

[46] Franklin, Gene F. and Powell, J. David, "Digital Control of Dynamic Systems", Addison-Wesley Publishing Company, 1981.

[47] Foster, I., “Designing and Building Parallel Programs”, Addison Wesley, 1995.

[48] Funahashi, K.I., "On the approximate realization of continuous mappings by neural networks", Neural Netwoks, Vol.2, pp.183-192, 1989.

[49] Fu-Sheng, H. and Ioannou, P., "Traffic Flow Modeling and Control using Artificial Neural Networks", IEEE Control Systems, October 1996.

[50] Fu, K.S., "Learning Control Systems and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control", Proceedings of the IEEE Transactions on Automatic Control, Vol.16, Technical Notes and Correspondence, pp.70-72, School Elect. Eng., Purdue University, Lafayette, Ind. 47907, February, 1971.

[51] Gardiner, Judith D. and Laub, Alan J., "Parallel Algorithms for Algebraic Riccati Equations", Int. J. Control, Vol.54, No.6, pp.1317–1333, 1991.

[52] Girosi, F. and T. Poggio, "Networks and the Best Approximation Property", Biological Cybernetics, Vol.63, No.3, pp.169-176, 1990.

[53] Golub, Gene H. and Loan, Charles F. Van, "Matrix Computations", The Johns Hopkins University,1989.

[54] C.J. Harris, "Advances in Intelligent Control", Taylor-Francis, 1994.

[55] Harvey, Andrew C., "Forescasting, Structural Time Series Models and the Kalman Filter", Cambridge University Press, 1992.

[56] Harvey, Andrew C., "Time Series Models", Harvester Wheatsheaf, 1993.

[57] Haykin S., "Neural Networks: A Comprehensive Foundation", Second Edition, Prentice Hall, 1999.

[58] Ho, B.L. and Kalman, R.E., "Effective Construction of Linear State-Variable Models from Input/Output Data", Proc. 3rd Annual Allerton Conf. on Circuit and Systems Theory, 449-459, 1965.

[59] Hornik K. and M. Stinchombe and H. White, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, Vol.2, pp.359-366, 1989.

[60] Hornik, K. et al., "Degree of Approximation Results for Feedforward Networks and Approximation Unknown Mapping and Their Derivatives", Neural Computation, Vol.6, No.6, pp.1262-1275, 1994.

[61] Hunt, L.R., DeGroat, R.D. and Linebarger, D.A., "Nonlinear AR modelling", Circuits Systems and Signal Processing 14, 689-705, 1995.

[62] Hwang, Kai and Xu, Zhiwei, "Scalable Parallel Computing", WCB/McGraw-Hill, 1998.

[63] Ieroham, S. Baruch, Jose-Martín Flores, Federico Thomas and Elena Gortcheva, "A Multimodel Recurrent Neural Network for Systems Identification and Control", IEEE International Joint Conference on Neural Networks, 2001.

[64] Ioannou,P.A. and J. Sun, "Robust adaptive Control", Prentice-Hall, 1996.

[65] Jagannathan, S., Vandegrift, M.W. and Lewis, F.L., "Adaptive Fuzzy Logic Control of Discrete-Time dynamical Systems", Automatica 36, pp. 229-241, 2000.

[66] James, T. Lo and Devasis, Bassu, "Adaptive vs Accommodative Neural Networks for Adaptive System Identification", IEEE International Joint Conference on Neural Networks, 2001.

[67] Jin-Tsong Jeng and Tsu-Tain Lee, "A Approximate Equivalence Neural Network to Conventional Neural Network for the Worst-Case Identification and control", IEEE International Joint Conference on Neural Networks, 1999.

[68] Juang, J.J., "Mathematical Correlation of Modal-Parameter-Identification Methods Via System-Realization Theory", The Int. Journal of Anal. and Experimental Modal Analysis, Vol.2, $N_0$1, January 1987.

[69] Juditsky, Anatoli, Hjalmarsson, H., Benveniste Alberto, Delyon Bernard and Ljung, Lennart, "Nonlinear Black-Box Models in System Identification: Mathematical Foundations", Automatica 31(12), 1725-1750, 1995.

[70] Kailath, Thomas, "Linear Systems", Prentice-Hall, Inc., 1980.

[71] Kalman, R.E., "On the General Theory of Control Systems", IFAC, International and Remote Control, Vol.1, pp.481-492, 1960.

[72] Klema, Virginia C. and Laub, Alan J., "The Singular Value Decomposition: Its Computation and some Applications", IEEE Trans. on Automatic Control, AC-25 (2), 164-176, April 1980.

[73] Korba, P., Babuska, R., H.B. Verbruggen and P.M. Frank, "Fuzzy Gain Scheduling: Controller and Observer Design based on Lyapunov method and Convex Optimization", IEEE Transaction on Fuzzy Systems, Vol.11, No.3, pp.285-298, 2003.

[74] Kristinsson, K. and GA. Dumont, "Genetic Algorithms in System Identification", Third IEEE Int. Symp. Intelligent Control, pp.597-602, 1988.

[75] Kristinsson, K. and GA. Dumont, "System Identification and Control Using Genetic Algorithm", IEEE Transactions on Systems Man and Cybernetics, Vol.22, No.5, pp.1033-1046, September/October, 1992.

[76] Kwakernaak, H and Sivan, R, "Linear Optimal Control Systems", Wiley-Interscience, New York-USA, 1972.

[77] Landau, ID., "Adaptive Control: The Model Reference Approach,Ed. Marcel Dekker, New York, 1979.

[78] Larimore, Wallace E., "Canonical Variate Analysis in Identification, Filtering and Adaptive Control", In Proc.29^{th} IEEE Conf. Decision and Control, pp.596-604, 1990.

[79] Laub, Alan J., "A Schur Method for Solving Algebraic Riccati Equations", IEEE Transactions on Automatic Control, December, V. AC-24, $N^0$6, 1979.

[80] Lawrence, D.A. and Rugh, W.J., "Gain Scheduling Dynamic Linear Controllers for a Nonlinear Plant", Automatica, 31(3), Pp.381-390, 1995.

[81] Lee, T.T., Jeng, J.T. and Shih, C.L., "Using Neural Networks to improve Gain Scheduling Techniques for Linear Parameter Varying Systems", Proc. Int. Workshop Advanced Motion Contr., Mar., 1996.

[82] Lee, L. and Poolla, K., "Identification of Linear Parameter-Varying Systems via LFTs", In Proc. of the 1996 Conference on Decision and Control, 1996.

[83] Lewis, F.L., "Optimal Control", John Wiley $\&$ Sons, 1986.

[84] Lewis, F.L. and Syrmos, V.L., "Optimal Control", Second edition, John Wiley $\&$ Sons, 1995.

[85] Lewis, F.L. and K.Liu and A.Yesilderik, "Multilayer Neural Robot Controller With Guaranteed Performance", IEEE Transaction on Neural Networks, Vol.6, No.3, pp.703-715, May, 1995.

[86] Lewis, F.L. and S.Jagannathan and A. Yesildirek, "Neural Network Control of Robot Manipulators and Nonlinear Systems", Taylor & Francis, 1999.

[87] Lin C.L., Chi-Chih Lai and Teng-Hsien Huang, "A Neural Network for Linear Matrix Inequality Problems", IEEE Transactions on Neural Networks, Vol.11, N_0 5, September 2000.

[88] Lin, J. and Lewis, F.L., "Two-Time Scale Fuzzy Logic Controller of Flexible Link Robot Arm", Fuzzy Sets and Systems 139, pp 125-149, 2003.

[89] Linkens, DA. and H.O. Nyongesa, "Learning Systems in Intelligent Control: an Appraisal of Fuzzy, Neural and Genetic Algorith Control Applications", IEEE Proc. Control Theory Appl., Vol.143, No.4, pp.367-386, July, 1996.

[90] Ljung, Lennart, "System Identification: Theory for the User", Prentice Hall, 1987.

[91] Ljung, Lennart and T. Söderström, "Theory and Practice of Recursive Identification", The MIT Press Cambridge, MA, 1983.

[92] Ljung, Lennart and Sjöberg, Jonas, "A System Identification Perspective on Neural Nets", Citeseer, http://citeseer.nj.nec.com, 1992.

[93] Lovera, Marco, "Subspace Identification Methods: theory and applications", Ph.D. Thesis, Politecnico di Milano, 1998.

[94] Lovera, Marco, Gustafsson, T. and Verhaegen, Michel, "Recursive Subspace Identification of Linear and Non-Linear Wiener State Space Models", Automatica 36, pp. 1639-1650, 2000.

[95] Madam, M. Gupta and Naresh K. Sinha, Editors, "Intelligent Control Systems–Theory and Applications", IEEE–Press, New York, USA, 1996.

[96] Maia, Carlos Andrey and Resende, Peterson, "Neural Control of MIMO Nonlinear Plants: a Gain Scheduling Approach", Proceedings of the IEEE International Symposium on Intelligent Control, pp.193-198, Istanbul, 1997.

[97] Maia, Carlos Andrey and Resende, Peterson, "Um Controlador Neural Gain Scheduling para Plantas Não-Lineares", SBA Controle & Automação, Vol.9, No.3, pp.135-140, 1998.

[98] Melin, Patricia and Oscar Castillo, "Intelligent Control of Non-Linear Plants using Type-2 Fuzzy Logic and Neural Networks", Proceedings IEEE, pp.1558-1562, 2003.

[99] Miyasato, "Adaptive Gain-Scheduled H_∞ Control of Linear Parameter-Varying Systems with Nonlinear Components", Proceedings of the American Control Conference, pp.208-213, June 4-6, Denver, Colorado, 2003.

[100] McNichols, K.H. and Fadali, S., "Selecting Operating Points for Discrete-time Gain scheduling", Computers & Electrical Engineering, Vol 29(2), March 2003, Pp. 289-301.

[101] Monett, D., T. Luis, A. Soto, Tamariz, A.D.R., H.-D. Burkhard and K. Bothe, "10 Años de Cooperación en el Tema: Sistemas Inteligentes", Memorias de Diálogos transatlânticos, *Anais da Conferencia Conjunta de la Universidad de La Habana y la Universidad de Humboldt de Berlín*, La Habana, Cuba, Feb-Mar, 2000.

[102] Narendra, K.S. and AM. Annaswamy, "Stable Adaptive System", Prentice Hall, Englewood Cliffs, NJ, 1989.

[103] Narendra, K.S. and K.Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Trans. on Neural Networks, 1, 4-27, 1992.

[104] Narendra, KS. and Snehasis Mukhopadhyay, "Intelligent Control Using Neural Networks", Proceedings of the IEEE, April, 1992.

[105] Narendra, K.S., "Neural Networks for Control: Theory and Practice", Proceedings of the IEEE, 84(10), 1385-1406, 1996.

[106] Ohsumi, Akira and Kawano T., "Subspace Identification for a Class of Time-Varying Continuous-Time Stochastic Systems Via Distribution Based Approach", Proceedings 15^{th} IFAC World Congress, Barcelona, July 2002.

[107] Oliveira, Mauricio C., Geromel, J.C., Liu Hsu, "LMI Characterization of Structural and Robust Stability", Linear Algebra and its Applications, (285), 69-80, 1998.

[108] Oliveira, Mauricio C., "Controle de Sistemas Lineares Baseado nas Desigualdades Matriciais Lineares", Tese de Doutorado, Faculdade de Engenharia Elétrica e Computação, UNICAMP, 1999.

[109] Oliveira, Mauricio C., Geromel, J.C., Liu Hsu; "LMI Characterization of Structural and Robust Stability: The Discrete-Time Case", Linear Algebra and its Applications, (296), 27-38, 1999.

[110] Oliveira, Mauricio C., Geromel, J.C., Liu Hsu, "Uma Formulação LMI para a Análise de Estabilidade com Funções de Lyapunov do Tipo Lure-Persidskii", Revista Controle & Automação, Vol.13, N_0 1, Jan-Abril 2002.

[111] Oliveira Serra, Ginalner Luiz and Bottura, C.P., "Neural Gain Scheduling Multiobjective Genetic Fuzzy PI Control", Proceedings of the IEEE International Symposium on Intelligent Control, ISIC, pp. 483-488, 2004.

[112] Packard, A., "Gain Scheduling via Linear Fractional Transformations", In Sys. Contr. Lett., Vol.22, pp.436-442, 1994.

[113] Panos, J. Antsaklis -Chair, J.S Albus, M.D Lemmon, K.M. Passino G.N. Saridis and P. Werbos, "Defining Intelligent Control - Report of the task force on intelligent Control", IEEE Control System Magazine, pp. 4-5,58-66, 1994.

[114] Park, J. and IW. Sandberg, "Universal Approximation Using Radial-Basis-Function networks", Neural Comp., Vol.3, pp.246-257, 1991.

[115] Patterson, D.A. and Hennessy, J.L., "Computer Organization $\&$ Design - The Hardware, Software Interface", Second Edition, Morgan Kaufmann Publishers, Inc. San Francisco, California, 1998.

[116] Polycarpou, M.M. and Ioannou, P.A., "Neural Networks and On-Line Approximators for Discrete-Time Nonlinear system Identification", Preprint, 1991.

[117] Polycarpou, MM. and Ioannou, PA., "Identification and Control Using Neural Networks Models: design and stability analysis", Dept. of Elec. Eng., Tech Report 91-09-01, September 1991.

[118] Polycarpou, M.M. and Ioannou, P.A., "Modeling, Identification and Stable Adaptive Control of Continuous-Time Nonlinear Dynamical Systems Using Neural Networks", Proceedings of ACC'92, 36-40, 1992.

[119] Polycarpou, M.M., "Stable adaptive neural control scheme for nonlinear systems, IEEE Transactions on Automatic Control, 41, 447-451, 1996.

[120] Poznyak, A.S. and Ljung, Lennart, "On-line Identification and Adaptive Trajectory Tracking for Nonlinear Stochastic Continuous Time Systems using Differential Neural Networks", Automatica 37, pp.1257-1268, 2001.

[121] Raol, J.R. and Madhuranath, H., "Neural network architectures for Parameter Estimation of Dynamical Systems", IEE Proc. Control Theory Appl., Vol. 143, N_o. 4, July 1996.

[122] Rovithakis, G.A., "Tracking control of multi input affine nonlinear dynamical systems with unknown Nonlinearities using Dynamical Neural Networks", IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 29, 179-189, 1999.

[123] Rovithakis, G.A., "Robust neural adaptive stabilization of unknown systems with measurement noise", IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 29, 453-458, 1999.

[124] Rovithakis, G.A., "Performance of a Neural Adaptive Tracking Controller for Multi Input Nonlinear Dynamical Systems in the Presence of Additive and Multiplicative External Disturbances", IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems ans Humans, 30, 720-730, 2000.

[125] Rovithakis, G.A., "Stable Adaptive Neuro Control Design via Lyapunov Function Derivative Estimation", Automatica 37, 1213-1221,2001.

[126] Rugh, W.J., “Analytical Framework for Gain Scheduling”, IEEE Control Systems Magazine, January, Pp.79-84,1991.

[127] Rugh, W.J. and J. Shamma, "Research on Gain Scheduling", Automatica 36, Pp.1401-1425, 2000.

[128] Sadegh, N., "A perceptron Network for Functional Identification and Control of Nonlinear Systems", IEEE Transaction on Neural Networks, Vol.4, No.6, pp.982-988, November, 1993.

[129] Sanner, RM. and JJ. Slotine, "Gaussian Networks for Direct Adaptive Control", IEEE Trans. on Neural Networks, Vol.3, No.6, pp.837-863, November, 1992.

[130] Shibata, T. and T. Fukuda, "Hierarchical Intelligent Control for Robotic Motion", IEEE Transaction on Neural Network, pp.823-832, September, 1994.

[131] Shibata, T., T. Abe, K.Tanie and M. Nose, "Skill-based Motion Planning in Hierarchical Intelligent Control of a Redundant Manipulator", Robotics Autom. Syst., Vol.18, pp.65-73, 1996.

[132] Scherer, Carsten, "The Riccati Inequality and State-Space H_∞-Optimal Control", PhD., 1990.

[133] Scherer, Carsten and Siep Weiland, "Linear Matrix Inequalities in Control", October 2000.

[134] Shamma, J.S., "Analysis and Design of Gain Scheduled Control Systems", PhD Thesis, MIT Report by NASA, 1988.

[135] Shaw, I.S. and M.G. Simões, "Controle e Modelagem Fuzzy", Editora Edgard Blucher LTDA., 1999.

[136] Shokoohi, S. and Silverman, L.M., "Identification and Model Reduction of Time-Varying Discrete-Time Systems", Automatica 23, pp. 509-522, 1987.

[137] Sugisaka, M., "A New Identification Method Using a Neurocomputer", SYSID 1997.

[138] Suykens, J.A.K., Joos, P.L. Vandewalle and DeMoor, B.L.R., "Artificial Neural Networks for Modelling and Control of Non-Linear Systems", Kluwer Academic Publishers, 1997.

[139] S*ö*derstr*ö*m, T. and Stoica, P., "System Identification", Prentice Hall, 1989.

[140] Swindlehurst, A., R.Roy, B. Ottersten and Kailath, T., "A Subspace Fitting Method for Identification of Linear State Space Models", IEEE Transactions on Automatic Control, 1995.

[141] Tamariz, A.D.R., Bottura, C.P., J.V. Fonseca and Barreto, G., "A Parallel and Distributed Solution Method for the Riccati Equation Via Stabilized Elementary Similarity Transformation", Ninth SIAM Conference on Parallel Processing for Scientific Computation, March 22-29, 1999.

[142] Tamariz, A.D.R., Bottura, C.P., Barreto, G. and M.J. Bordon, "A High Computational Performance Approach for a Subspace Identification Method", *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, (SIAM'99), San Antonio, Texas, March 22-24, 1999.

[143] Tamariz, A.D.R., "Uma Proposta Metodologica Para Solução da Equação Algébrica de Riccati em Formas Sequencial e Paralela", Tese de Mestrado, Faculdade de Engenharia Elétrica e Computação, UNICAMP, Abril, 1999.

[144] Tamariz, A.D.R., Bottura, C.P., J.V. Fonseca and Barreto, G., "Formas Sequencial e Paralela para Solução da Equação Algébrica de Riccati por um Algoritmo de Schur-Modificado", *Anais do XV Congresso Brasileiro de Engenharia Mecânica*, (COBEM'99), Águas de Lindóia, São Paulo, Brasil, 22-26 Novembro, 1999.

[145] Tamariz, A.D.R., Bottura, C.P., J.V. Fonseca and Barreto, G., "Parallel and Distributed Multivariable Identification Via the MOESP Approach, *Anais do XVI Congresso Brasileiro de Engenharia Mecânica*, (COBEM'01), Uberlândia, MG, 26-30 novembro, 2001.

[146] Tamariz A.D.R., Bottura C.P. and Barreto G., "Algoritmo Iterativo do tipo MOESP para Identificação de sistemas discretos variantes no tempo - Parte I: Formulação", DINCON, 2003.

[147] Tamariz A.D.R., Bottura C.P. and Barreto G., "Algoritmo Iterativo do tipo MOESP para Identificação de sistemas discretos variantes no tempo - Parte II: Implementação e Experimentação", DINCON, 2003.

[148] Tamariz, A.D.R., Bottura, C.P. and Barreto, G., "Discrete Time Variant System Identification", *Anais do* 1^{st} *Meeting on Computational Modelling (LNCC)*, Petrópolis, August 9-13, 2004.

[149] Tamariz, A.D.R. and Bottura, C.P., "Discrete-Time Systems Neuro-Riccati Equation Solution", IEEE International Joint Conference on Neural Networks, IJCNN-2005, August, Quebec, Montreal, Canada, 2005.

[150] Tamariz, A.D.R. and Bottura, C.P., "Discrete-Time Algebraic Riccati Inequation Neuro-LMI Solution", *Proceedings of the International Conference on Systems, Man and Cybernetics (IEEE SMC 2005)*, Hawaii-USA, October 10-12 2005.

[151] Tamariz, A.D.R. and Bottura, C.P. and Barreto, G., "Iterative MOESP Type Algorithm for Discrete Time Variant System Identification", 13th Mediterranean Conference on Control and Automation, (MED'2005), 27-29 Junho, Limassol, Cyprus, 2005.

[152] Tamariz, A.D.R. and Bottura, C.P. and Serra, G., "Intelligent Gain-Scheduling Control for Multivariable Discrete Linear Time Varying Systems", XVIII Congresso Brasileiro de Engenharia Mecânica, COBEM'05, 6-11 novembro, Ouro Preto, MG, Brasil, 2005.

[153] Tamariz, A.D.R. and Bottura, CP., "Soluções Neurais de Equações Algébricas de Riccati", *VII Congresso Brasileiro de Redes Neurais (CBRN 2005)*, Natal, 16 a 19 de outubro de 2005.

[154] Tamariz, A.D.R. and Bottura, CP., "Soluções Neurais de Inequações Matriciais Lineares de Riccati", *VII Simpósio Brasileiro de Automação Inteligente (SBAI 2005)*, São Luís, Maranhão-Brasil, 19-23 Setembro, 2005.

[155] Terasvirta, T., Lin, C.F. and Granger, W.J., "Power of the Neural Network linearity test, J. Time Ser. Anal. 14-2, 209-220, 1993.

[156] Tong, H., "Nonlinear Time Series: A dynamical system approach, Oxford University Press, New York, 1990.

[157] Van Huffel, Sabine and Vandewalle, Joos, "The Total Least Squares Problem: Computational Aspects and Analysis", Frontiers in Applied Mathematics, SIAM, Philadelphia, 1991.

[158] Van Overschee, Peter and De Moor, Bart, "N4SID: Subspace algorithms for the Identification of combined deterministic-stochastic systems", Automatica, Special Issue on statistical Signal Processing and Control, 30(1):75-93, 1994.

[159] Van Overschee, Peter and De Moor, Bart, "Subspace Identification of Linear Systems: Theory, Implementation, Applications", Kluwer, Academic Publishers, 1996.

[160] Vaughan D.R., "A Nonrecursive Algebraic Solution for the Discrete Riccati Equation", IEEE Transactions on Automatic Control, October, Vol.AC-15, number 5, 597–599, 1970.

[161] Verhaegen, Michel and Deprettere, E., "A fast recursive MIMO State Space model Identification Algorithm", Proceedings of the 30th Conference on Decision and Control, Brighton, England, December 1991.

[162] Verhaegen, Michel and Dewilde, Patrick., “Subspace Model Identification - Part 1 : The output-error state-space model identification class of algorithms”, International Journal of Control, Volume 56, Number 5, 1187-1210, November, 1992.

[163] Verhaegen, Michel and Dewilde, Patrick., “Subspace Model Identification - Part 2 : Analysis of the Elementary Output-Error State-Space Model Identification Algorithms", International Journal of Control, Vol.56, No.5, pp.1211-1241, November, 1992.

[164] Verhaegen, Michel, "Subspace Model Identification - Part 3 : Analysis of the Ordinary Output-Error State-Space Model Identification Algorithms", International Journal of Control, Vol.58, No.3, pp.555-586, September, 1993.

[165] Verhaegen, Michel and Yu Xiaode, "A Class of Subspace Model Identification Algorithms to Identify Periodically and Arbitrarily Time-varying Systems", Automatica, Vol. 31, N_0. 2, pp. 201-216, 1995.

[166] Viberg M., B. Wahlberg and B. Ottersten, “Analysis of State Space System Identification Methods Based on Instrumental Variables and Subspace Fitting”, Automatica, Volume 33, number 9, pages 1603-1616, 1997.

[167] Xia, Y., Wang, J. and D.L. Hung, "Recurrent Neural Networks for solving Linear Inequalities and Equations", IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol.46, N_0 4, April 1999.

[168] Xinmin, Dong, Xiong Zhiguo and Liu Quin, "Gain Scheduled model following control of flight control system based on Neural Network", IEEE Int. Conference on Neural Networks $\&$ Signal Processing, pp.301-305, December 14-17, 2003.

[169] Wang, J., "Recurrent Neural Networks for solving Linear matrix equations, Computers $\&$ Mathematical with applications, Vol.26, No. 9, Pp. 23 - 34, 1993.

[170] Wang, J., "Solving Simultaneous Linear equations using Recurrent Neural Networks, Information Sciences, Vol.76, No. 3-4, Pp. 255 - 277, 1994.

[171] Wang, J. and Guang, Wu, "A Multilayer recurrent neural network for solving continuous-time algebraic Riccati equations, Neural Networks, 11, Pp. 939 - 950, 1998.

[172] Zhang, Y., Jiang, D. and Wang, J., "A Recurrent Neural Network for solving Sylvester Equation with Time-Varying coefficients", IEEE Transactions on Neural Networks, Vol.13, $N_0$5, September 2002.

[173] Zhou K, Doyle JC and Glover K, "Robust and Optimal Control", Prentice Hall, 1996.

Capítulo 8

Apêndices

8.1 Derivadas

Sejam duas matrizes $A, B \in \Re^{n \times n}$, vamos apresentar algumas propriedades das matrizes estudadas e muito utilizadas nesta teses, na especificação teórica da definição das equações dinâmicas neurais utilizadas na implementação das Redes Neurais.

- Derivada da multiplicação de matrizes:

$$\frac{d[A(t)B(t)]}{dt} = \dot{A}(t)B(t) + A(t)\dot{B}(t)$$

- Derivada de matriz exponencial:

$$\frac{dA^2(t)}{dt} = \dot{A}(t)A(t) + A(t)\dot{A}(t)$$

- Derivada da soma de matrizes:

$$\frac{d[A(t) + B(t)]}{dt} = \dot{A}(t) + \dot{B}(t)$$

- Derivada da inversa de uma matriz:

$$\frac{dA^{-1}(t)}{dt} = -A^{-1}(t)\dot{A}(t)A^{-1}(t)$$

  onde $\dot{A}(t) = \sum_{ij} \dot{a}_{ij}$, ou seja corresponde com a derivada de cada elemento da matriz A.

A seguir apresentamos o desenvolvimento de algumas derivadas de matrizes as quais trabalhamos neste projeto, para uma melhor compreensão do trabalho.

1. $E = A^T X$

$$\frac{\partial E}{\partial X} = \begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \cdots & \frac{\partial E}{\partial x_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial x_{n1}} & \cdots & \frac{\partial E}{\partial x_{nn}} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n}\sum_{l=1}^{n} f_{kl}(g)\frac{\partial \sum_{i=1}^{n} a_{ik}x_{il}}{x_{11}} & \cdots & \sum_{k=1}^{n}\sum_{l=1}^{n} f_{kl}(g)\frac{\partial \sum_{i=1}^{n} a_{ik}x_{il}}{x_{1n}} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n}\sum_{l=1}^{n} f_{kl}(g)\frac{\partial \sum_{i=1}^{n} a_{ik}x_{il}}{x_{n1}} & \cdots & \sum_{k=1}^{n}\sum_{l=1}^{n} f_{kl}(g)\frac{\partial \sum_{i=1}^{n} a_{ik}x_{il}}{x_{nn}} \end{bmatrix} \tag{8.1}$$

Desenvolvendo cada termo da matriz acima, obtemos por exemplo que o termo $(1,1)$ da matriz resultante é igual a:

$$\sum_{k=1}^{n} f_{k1}(g)\frac{\partial(a_{1k}x_{11} + a_{2k}x_{21})}{\partial x_{11}} + \sum_{k=1}^{n} f_{k2}(g)\frac{\partial(a_{1k}x_{12} + a_{2k}x_{22})}{\partial x_{11}} = \sum_{k=1}^{n} f_{k1}(g)a_{1k}$$

Continuando com o processo acima, temos que a derivada da função de enêrgia $E = A^T X$ em relação a X é igual

$$\frac{\partial A^T X}{\partial X} = \begin{bmatrix} \sum_{k=1}^{n} f_{k1}(g)a_{1k} & \cdots & \sum_{k=1}^{n} f_{k1}(g)a_{nk} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} f_{kn}(g)a_{1k} & \cdots & \sum_{k=1}^{n} f_{kn}(g)a_{nk} \end{bmatrix} = F^T A^T$$

8.2 Formas quadráticas definidas

Seja $Q \in \mathcal{R}^{n\times n}, Q = Q^T$. Então

1. Q é definida positiva (negativa) se $x^T Q x > (<) 0$ para todo $x \in \mathcal{R}^n, x \neq 0$; ou se e somente se (sse) todos os autovalores de Q são positivos (negativos).

2. Q é semi-definida positiva (negativa) se $x^T Q x \geq (\leq) 0$ para todo $x \in \mathcal{R}^n$; ou sse todos os autovalores de Q são não-negativos (não-positivos).

3. Q é indefinida se existem vetores $x, y \in \mathcal{R}^n$ tais que $x^T Q x < 0 < y^T Q y$; ou sse Q possui ao mesmo tempo autovalores positivos e negativos.

8.2.1 Cálculo diferencial aplicado a formas quadráticas

Dados $x \in \Re^n$, $y \in \Re^n$ e $A \in \Re^{n\times n}$:

- $$\frac{\partial(y^T Ax)}{\partial y} = Ax$$

- $$y^T Ax = x^T A^T y \Longrightarrow \frac{\partial(y^T Ax)}{\partial x} = \frac{\partial(x^T A^T y)}{\partial x} = A^T y$$

- $$\frac{\partial(x^T Ax)}{\partial x} = A^T x + Ax$$

  para

  $$A^T = A,\ \frac{\partial(x^T Ax)}{\partial x} = 2Ax$$

8.3 Definições

Definição 1 *Uma matriz infinita H bloco Hankel de uma matriz resposta ao ruído branco $\begin{bmatrix} L_{k,j} & G_{k,j} \end{bmatrix}$ é:*

$$H = \begin{bmatrix} \begin{bmatrix} L_1 & G_1 \\ L_2 & G_2 \end{bmatrix} & \begin{bmatrix} L_2 & G_2 \\ L_3 & G_3 \end{bmatrix} & \begin{bmatrix} L_3 & G_3 \\ L_4 & G_4 \end{bmatrix} & \cdots \\ \begin{bmatrix} L_3 & G_3 \end{bmatrix} & \begin{bmatrix} L_4 & G_4 \end{bmatrix} & \ddots & \\ \vdots & & & \end{bmatrix} \tag{8.2}$$

Teorema 3 *Um processo estocástico gaussiano de média nula $\{x_t\}$ é markoviano se e somente se, qualquer uma das equivalências seguintes são verificadas:*

- $E(x_{n+k}|x_k, x_{k-1}, \ldots) = E(x_{n+k}|x_k)$ *para* $n \geq 0$
- $E(x_t|x_{t_2}, x_{t_1}) = E(x_t|x_{t_2})$ *para todo* t, t_1, t_2 *tal que* $t > t_2 > t_1$
- $\Phi(t, t_2) = \Phi(t, t_1)\Phi(t_1, t_2)$ *para todo* t, t_1, t_2 *tal que* $t > t_2 > t_1$

Prova 3 *A prova deste teorema pode ser vista em [32, 44].*

Teorema 4 *[32] Um processo estocástico gaussiano $\{x_t\}$ de média nula e covariância definida positiva $\Pi_{t,t}$, é um processo markoviano se e somente se, é verificada a seguinte equação linear estocástica a diferenças*

$$x_{t+1} = A_t x_t + v_t \tag{8.3}$$

onde $A_t = \Pi_{t+1,t}\Pi_{t,t}^{-1}$ e $v_t = x_{t+1} - A_t x_t$ é um processo ruído branco de covariâncias dadas por

$$E(v_t v_s^T) = \begin{cases} 0 & t \neq s \\ Q_t = \Phi_{t+1,t+1} - A_t \Phi_{t,t} A_t^T & t = s \end{cases} \tag{8.4}$$

Prova 4 *Se x_t é markoviano, então pelo Teorema 3 temos*

$$E(x_{t+1}|x_t, x_{t-1}, \ldots) = E(x_{t+1}|x_t) = \Pi_{t+1,t}\Pi_{t,t}^{-1}x_t = A_t x_t \tag{8.5}$$

daí segue que

$$x_{t+1} = A_t x_t + v_t \tag{8.6}$$

onde $v_t = x_{t+1} - A_t x_t$ é independente não só de x_t bem como de $x_{t-1}, x_{t-2}, \ldots$ e v_{t-1} é independente de $x_{t-1}, x_{t-2}, \ldots$ e assim sucessivamente.

O processo v_t é um processo ruído branco gaussiano.

$$\begin{aligned} E(v_t v_t^T) &= E\{(x_{t+1} - A_t x_t)(x_{t+1} - A_t x_t)^{-1}\} \\ &= E(x_{t+1}x_{t+1}^T) - E(x_{t+1}x_t^T)A_t^T - A_t E(x_t x_{t+1}^T) + A_t E(x_t x_t^T)A_t^T \\ &= E(x_{t+1}x_{t+1}^T) - E\{(A_t x_t + v_t)x_t^T\}A_t^T - A_t E\{x_t(A_t x_t + v_t)^T\} + A_t E(x_t x_t^T)A_t^T \\ &= E(x_{t+1}x_{t+1}^T) - A_t E(x_t x_t^T)A_t^T \end{aligned}$$

($\Leftarrow$) *suficiência*

$$\begin{aligned} E(A_t x_t + v_t|x_t, x_{t-1}, x_{t-2}, \ldots) &= E(A_t x_t|x_t, x_{t-1}, x_{t-2}, \ldots) + E(v_t|x_t, x_{t-1}, x_{t-2}, \ldots) \\ &= A_t x_t + 0 = E(x_{t+1}|x_t) \end{aligned}$$

Definição 2 *Chamamos de realização Markoviana (se existe) de um processo vetorial gaussianos estacionário y_k de média nula a um modelo da forma*

$$\left\{ \begin{array}{lcl} x_{k+1} &=& \mathbf{A}x_k + v_k \\ y_k &=& \mathbf{C}x_k + w_k \end{array} \right. \tag{8.7}$$

onde $\left[\begin{array}{c} v \\ w \end{array} \right]$ é um processo vetorial ruído branco gaussiano de média nula com covariância dada por:

$$E\left[\left(\begin{array}{c} v_k \\ w_k \end{array} \right) \left(\begin{array}{cc} v_s^T & w_s^T \end{array} \right) \right] = \left(\begin{array}{cc} Q & S \\ S^T & R \end{array} \right) \delta_{k,s} \tag{8.8}$$

onde $\delta_{k,s}$ é o delta de Kronecker, cujo valor esta definido por:

$$\delta_{k,s} = \left\{ \begin{array}{lll} 1 & se & k = s \\ 0 & se & k \neq s \end{array} \right. \tag{8.9}$$

e as matrizes A, C satisfazem às seguintes hipóteses:

- *A é uma matriz assintoticamente estável;*
- *O par (A, L) é completamente atingível, onde $Q = LL^T$;*
- *O par (A, C) é completamente observável.*
- A matriz real $A \in \Re^{n \times n}$ é Schur estável, se todos seus autovalores estão localizados no interior do círculo unitário no plano complexo. É bem conhecido que a matriz A é Schur se e somente se, existem duas matrices simétricas e definidas positivas P e Q que satisfazem a restrição linear $A^T PA - P + Q = 0$, denominada equação de Lyapunov Discreta no tempo.

8.4 Dados dos Experimentos

Nesta seção apresentamos as matrizes de dados dos diferentes exemplos apresentados, assim como os gráficos das simulações efetuadas no desenvolvimento dos Capítulos 2, 3 e 4 deste trabalho, para possível análise.

$$
Y = \begin{bmatrix}
-0.3676 & -2.0176 & -3.1705 & 0.3898 & -2.4571 & -3.9234 & 0.6335 & -2.0071 & 1.3667 & 2.3918 \\
-0.6845 & 2.2561 & -2.2704 & -0.1523 & 0.5745 & -1.6835 & -4.0191 & 1.1345 & 0.8998 & -2.0425 \\
5.0358 & -2.9261 & 2.2928 & -1.5666 & -0.4375 & 1.3810 & 1.0718 & 1.8594 & 0.6892 & -2.7077 \\
1.6704 & -0.4057 & 2.1877 & 0.5361 & 1.3749 & -2.5618 & -1.7185 & 0.4727 & 0.2427 & -1.9482 \\
-1.6350 & -1.6677 & 1.0783 & -0.8417 & 1.0528 & 0.8998 & -1.0574 & 1.6165 & 0.0101 & 1.1490 \\
-0.3268 & -0.9287 & 3.7048 & -1.0079 & 0.4031 & 2.6358 & 0.3329 & -0.4826 & 0.3944 & 2.5277 \\
-3.0052 & 2.3214 & 2.3247 & 1.7116 & -1.0660 & -1.5205 & -1.0087 & -1.3072 & 4.4641 & -4.5668 \\
-3.1756 & 2.0792 & 0.5096 & -0.6112 & 2.4966 & -2.3370 & -0.2053 & -2.3429 & 3.6864 & -0.6528 \\
0.3778 & -2.2980 & 2.4658 & -1.0495 & -6.5630 & 1.8159 & -0.3071 & 0.1252 & -0.0462 & -3.2254 \\
0.4351 & -1.7738 & 2.0500 & -0.3810 & -2.6862 & -2.3109 & -0.0593 & 2.7027 & 0.5573 & 1.4242 \\
-0.0961 & -1.1820 & -1.2713 & -0.4473 & -0.2758 & 6.6602 & 1.4940 & -0.2294 & 2.0501 & -0.0645 \\
2.1419 & 1.9292 & -0.9954 & 6.7297 & -5.1456 & 1.8825 & 1.7231 & -1.3733 & -0.4200 & -0.1621 \\
0.9896 & -2.2349 & 1.0406 & 5.4326 & -2.0762 & 1.1945 & 1.8170 & 2.7490 & 1.6709 & 2.4434 \\
5.4888 & -2.4714 & -7.1995 & 1.3842 & -2.0804 & -3.2043 & 2.2890 & -6.4004 & 4.6882 & -5.5588 \\
4.2043 & 1.4024 & -1.7940 & -1.4981 & -0.5454 & -2.6642 & -0.8089 & 2.4481 & -2.8870 & -3.1898 \\
0.1248 & 1.7699 & -2.8063 & -2.4585 & 1.3016 & -0.1681 & -2.4371 & 2.5815 & 3.6754 & -2.6852 \\
-3.9154 & 6.0176 & -0.6265 & 1.3649 & 0.1910 & 1.1385 & -2.1878 & -0.6160 & -0.0842 & 4.3025 \\
-4.5080 & 5.1061 & -5.0003 & 2.4506 & 1.9995 & 2.5391 & -3.0263 & 3.6531 & -1.3758 & 4.3752 \\
-0.7639 & -0.1653 & -1.4705 & 0.7315 & -2.5414 & -6.4491 & -3.6875 & -1.7396 & 2.9155 & 0.9887 \\
0.5076 & -0.7533 & 0.4404 & -2.2546 & 3.8985 & -0.5648 & -2.6458 & -2.9113 & 4.4327 & -1.5688
\end{bmatrix} \tag{8.10}
$$

$$
U_k = \left[\begin{array}{cccccccccc}
0.3273 & 0.7258 & -0.1364 & 0.0593 & 0.2944 & 1.6236 & 1.2540 & 0.5711 & 0.8156 & 0.6686 \\
0.1746 & -0.5883 & 0.1139 & -0.0956 & -1.3362 & -0.6918 & -1.5937 & -0.3999 & 0.7119 & 1.1908 \\
-0.1867 & 2.1832 & 1.0668 & -0.8323 & 0.7143 & 0.8580 & -1.4410 & 0.6900 & 1.2902 & -1.2025 \\
 & & & & & & & & & \\
-0.0198 & 0.2573 & -0.8051 & -0.9219 & -1.0106 & 1.6924 & 0.3803 & -0.0482 & 1.0950 & 0.8956 \\
-0.1567 & -1.0565 & 0.5287 & -2.1707 & 0.6145 & 0.5913 & -1.0091 & 0.0000 & -1.8740 & 0.7310 \\
-1.6041 & 1.4151 & 0.2193 & -0.0592 & 0.5077 & -0.6436 & -0.0195 & -0.3179 & 0.4282 & 0.5779 \\
 & & & & & & & & & \\
0.0403 & -0.2556 & -1.4751 & 0.3148 & 0.6232 & -0.9921 & -1.0078 & -0.1315 & -0.6355 & -0.9499 \\
0.6771 & -0.3775 & -0.2340 & 1.4435 & 0.7990 & 0.2120 & -0.7420 & 0.3899 & -0.5596 & 0.7812 \\
0.5689 & -0.2959 & 0.1184 & -0.3510 & 0.9409 & 0.2379 & 1.0823 & 0.0880 & 0.4437 & 0.5690 \\
 & & & & & & & & & \\
-0.8217 & -2.2023 & 0.3274 & -1.0039 & -1.1859 & 0.0557 & -1.1283 & 0.9535 & -1.1678 & -1.2132 \\
-0.2656 & 0.9863 & 0.2341 & -0.9471 & -1.0559 & -1.2173 & -1.3493 & 0.1286 & -0.4606 & -1.3194 \\
-1.1878 & -0.5186 & 0.0215 & -0.3744 & 1.4725 & -0.0412 & -0.2611 & 0.6565 & -0.2624 & 0.9312 \\
 & & & & & & & & & \\
0.0112 & 0.2316 & 0.2895 & -0.6841 & -0.3306 & 1.4885 & -0.2463 & -1.2013 & 0.4853 & -0.4348 \\
-0.6451 & -0.9898 & 1.4789 & -1.2919 & -0.8436 & -0.5465 & 0.6630 & -0.1199 & -0.5955 & -0.0793 \\
0.8057 & 1.3396 & 1.1380 & -0.0729 & 0.4978 & -0.8468 & -0.8542 & -0.0653 & -0.1497 & 1.5352 \\
 & & & & & & & & & \\
-0.6065 & -0.9036 & 0.5354 & -2.0543 & 1.0184 & -0.6817 & 0.2888 & -0.3679 & 0.7283 & -1.0226 \\
-1.3474 & 0.0359 & 0.5529 & 0.1326 & -1.5804 & -1.0246 & -0.4293 & -0.4650 & 2.1122 & 1.0378 \\
0.4694 & -0.6275 & -0.2037 & 1.5929 & -0.0787 & -1.2344 & 0.0558 & 0.3710 & -1.3573 & -0.3898 \\
 & & & & & & & & & \\
-1.3813 & 0.7079 & 1.8645 & -0.2111 & 0.6353 & -1.0998 & -0.4931 & 1.2366 & -1.2316 & 0.3792 \\
0.3155 & 1.9574 & -0.3398 & 1.1902 & -0.6014 & 0.0860 & 0.4620 & -0.6313 & 1.0556 & 0.9442 \\
1.5532 & 0.5045 & -1.1398 & -1.1162 & 0.5512 & -2.0046 & -0.3210 & -2.3252 & -0.1132 & -2.1204 \\
 & & & & & & & & & \\
-0.6447 & -0.1821 & 1.2274 & -0.7829 & 0.4801 & 0.8892 & -0.0118 & -1.1071 & -0.2762 & -0.5226 \\
-0.7043 & 1.5210 & -0.6962 & 0.5869 & 0.6682 & 2.3093 & 0.9131 & 0.4855 & 1.2765 & 0.1034 \\
-1.0181 & -0.0384 & 0.0075 & -0.2512 & -0.0783 & 0.5246 & 0.0559 & -0.0050 & 1.8634 & -0.8076 \\
 & & & & & & & & & \\
0.6804 & 0.2189 & -0.2747 & -1.6636 & -0.5412 & -0.7121 & -0.2494 & -1.6640 & -1.2566 & -1.1746 \\
-2.3646 & 0.2617 & -0.1331 & -0.7036 & -1.3335 & -0.0113 & 0.3966 & -1.0290 & -0.3472 & -1.0211 \\
0.9901 & 1.2134 & -1.2705 & 0.2809 & 1.0727 & -0.0008 & -0.2640 & 0.2431 & -0.9414 & -0.4017 \\
 & & & & & & & & & \\
0.1737 & -0.2454 & 0.0714 & 1.2781 & -0.0132 & -0.2576 & 0.3255 & 1.2698 & -0.1390 & -0.0154 \\
-0.1161 & -1.5175 & 0.3165 & -0.5478 & -0.5803 & -1.4095 & -1.1190 & -0.8960 & -1.1634 & 0.5362 \\
1.0641 & 0.0097 & 0.4998 & 0.2608 & 2.1363 & 1.7701 & 0.6204 & 0.1352 & 1.1837 & -0.7164
\end{array}\right] \tag{8.11}
$$

$$U_k = \begin{bmatrix}
-0.4326 & 0.2877 & 1.1892 & 0.1746 & -0.5883 & 0.1139 & -0.0956 & -1.3362 & -0.6918 & -1.5937 \\
-1.6656 & -1.1465 & -0.0376 & -0.1867 & 2.1832 & 1.0668 & -0.8323 & 0.7143 & 0.8580 & -1.4410 \\
0.1253 & 1.1909 & 0.3273 & 0.7258 & -0.1364 & 0.0593 & 0.2944 & 1.6236 & 1.2540 & 0.5711 \\
\\
-0.3999 & 0.7119 & 1.1908 & -0.1567 & -1.0565 & 0.5287 & -2.1707 & 0.6145 & 0.5913 & -1.0091 \\
0.6900 & 1.2902 & -1.2025 & -1.6041 & 1.4151 & 0.2193 & -0.0592 & 0.5077 & -0.6436 & -0.0195 \\
0.8156 & 0.6686 & -0.0198 & 0.2573 & -0.8051 & -0.9219 & -1.0106 & 1.6924 & 0.3803 & -0.0482 \\
\\
0.0000 & -1.8740 & 0.7310 & 0.6771 & -0.3775 & -0.2340 & 1.4435 & 0.7990 & 0.2120 & -0.7420 \\
-0.3179 & 0.4282 & 0.5779 & 0.5689 & -0.2959 & 0.1184 & -0.3510 & 0.9409 & 0.2379 & 1.0823 \\
1.0950 & 0.8956 & 0.0403 & -0.2556 & -1.4751 & 0.3148 & 0.6232 & -0.9921 & -1.0078 & -0.1315 \\
\\
0.3899 & -0.5596 & 0.7812 & -0.2656 & 0.9863 & 0.2341 & -0.9471 & -1.0559 & -1.2173 & -1.3493 \\
0.0880 & 0.4437 & 0.5690 & -1.1878 & -0.5186 & 0.0215 & -0.3744 & 1.4725 & -0.0412 & -0.2611 \\
-0.6355 & -0.9499 & -0.8217 & -2.2023 & 0.3274 & -1.0039 & -1.1859 & 0.0557 & -1.1283 & 0.9535 \\
\\
0.1286 & -0.4606 & -1.3194 & -0.6451 & -0.9898 & 1.4789 & -1.2919 & -0.8436 & -0.5465 & 0.6630 \\
0.6565 & -0.2624 & 0.9312 & 0.8057 & 1.3396 & 1.1380 & -0.0729 & 0.4978 & -0.8468 & -0.8542 \\
-1.1678 & -1.2132 & 0.0112 & 0.2316 & 0.2895 & -0.6841 & -0.3306 & 1.4885 & -0.2463 & -1.2013 \\
\\
-0.1199 & -0.5955 & -0.0793 & -1.3474 & 0.0359 & 0.5529 & 0.1326 & -1.5804 & -1.0246 & -0.4293 \\
-0.0653 & -0.1497 & 1.5352 & 0.4694 & -0.6275 & -0.2037 & 1.5929 & -0.0787 & -1.2344 & 0.0558 \\
0.4853 & -0.4348 & -0.6065 & -0.9036 & 0.5354 & -2.0543 & 1.0184 & -0.6817 & 0.2888 & -0.3679 \\
\\
-0.4650 & 2.1122 & 1.0378 & 0.3155 & 1.9574 & -0.3398 & 1.1902 & -0.6014 & 0.0860 & 0.4620 \\
0.3710 & -1.3573 & -0.3898 & 1.5532 & 0.5045 & -1.1398 & -1.1162 & 0.5512 & -2.0046 & -0.3210 \\
0.7283 & -1.0226 & -1.3813 & 0.7079 & 1.8645 & -0.2111 & 0.6353 & -1.0998 & -0.4931 & 1.2366 \\
\\
-0.6313 & 1.0556 & 0.9442 & -0.7043 & 1.5210 & -0.6962 & 0.5869 & 0.6682 & 2.3093 & 0.9131 \\
-2.3252 & -0.1132 & -2.1204 & -1.0181 & -0.0384 & 0.0075 & -0.2512 & -0.0783 & 0.5246 & 0.0559 \\
-1.2316 & 0.3792 & -0.6447 & -0.1821 & 1.2274 & -0.7829 & 0.4801 & 0.8892 & -0.0118 & -1.1071 \\
\\
0.4855 & 1.2765 & 0.1034 & -2.3646 & 0.2617 & -0.1331 & -0.7036 & -1.3335 & -0.0113 & 0.3966 \\
-0.0050 & 1.8634 & -0.8076 & 0.9901 & 1.2134 & -1.2705 & 0.2809 & 1.0727 & -0.0008 & -0.2640 \\
-0.2762 & -0.5226 & 0.6804 & 0.2189 & -0.2747 & -1.6636 & -0.5412 & -0.7121 & -0.2494 & -1.6640 \\
\\
-1.0290 & -0.3472 & -1.0211 & -0.1161 & -1.5175 & 0.3165 & -0.5478 & -0.5803 & -1.4095 & -1.1190 \\
0.2431 & -0.9414 & -0.4017 & 1.0641 & 0.0097 & 0.4998 & 0.2608 & 2.1363 & 1.7701 & 0.6204 \\
-1.2566 & -1.1746 & 0.1737 & -0.2454 & 0.0714 & 1.2781 & -0.0132 & -0.2576 & 0.3255 & 1.2698
\end{bmatrix} \tag{8.12}$$

$$
Yk = \left[\begin{array}{cccccccccc}
-0.1653 & -1.8069 & -0.8584 & -1.3400 & 0.5955 & -0.1692 & -0.3542 & -1.6709 & -1.8194 & -1.8441 \\
0.3905 & 0.7455 & -1.8343 & 0.3891 & 1.3794 & 0.1053 & 0.1311 & 4.0663 & 1.6485 & 1.2491 \\
\\
-1.8873 & -0.9723 & -1.5877 & -0.8569 & 1.6201 & 1.2179 & 3.0095 & -1.9651 & -0.3601 & -1.4684 \\
1.0586 & -0.6008 & -3.4635 & -0.0734 & 1.4274 & -1.3895 & 3.2300 & 1.7323 & -1.5438 & 0.4177 \\
\\
-1.0162 & -0.6272 & -1.6816 & 0.1049 & 1.6845 & 0.1929 & -0.5292 & 0.2360 & 1.2874 & 1.4557 \\
1.3703 & 3.5757 & -2.1554 & -1.4816 & -1.3419 & 1.6810 & -1.5801 & -2.9484 & -1.0912 & 2.2062 \\
\\
0.8132 & 1.3908 & 1.1138 & 3.2498 & -0.2840 & 2.7171 & 1.0468 & 1.8438 & 2.1168 & -0.9475 \\
-1.4302 & -0.0074 & -1.7046 & -2.1119 & -0.1236 & -0.9938 & 0.1185 & 3.4495 & 0.4965 & 3.3991 \\
\\
1.5802 & 0.9547 & 1.7088 & 0.1269 & -0.4376 & -0.5651 & 0.7956 & -1.9170 & -0.3045 & -0.6882 \\
-1.9875 & -0.9109 & 3.3143 & 1.1853 & 1.4021 & -4.1592 & 1.2372 & 2.6982 & -1.2517 & -3.8569 \\
\\
0.6083 & 0.7559 & 0.3479 & 2.3900 & -1.0558 & 2.3986 & -1.0129 & 2.5123 & -2.0953 & 1.4418 \\
1.6409 & 0.0289 & -0.3761 & 1.5276 & 0.3195 & -3.7897 & 2.1811 & 1.3881 & 0.7222 & 0.4389 \\
\\
-1.1819 & 0.0395 & 0.9792 & 0.7966 & -2.5474 & -1.0865 & -2.7494 & 2.6257 & -0.0689 & -0.2041 \\
1.3982 & -5.7650 & -3.1034 & 2.2568 & -1.2081 & -1.7416 & -1.9456 & 0.3238 & -0.7867 & 2.0330 \\
\\
1.5039 & -1.2446 & 2.3606 & 0.0261 & -0.3909 & 1.5905 & -1.2000 & 0.2067 & -1.5459 & 1.4812 \\
-1.2760 & -0.2374 & -1.5430 & 1.5716 & 0.6252 & 0.4127 & 0.3010 & 0.9623 & -3.9796 & -2.2203 \\
\\
0.6906 & 1.6147 & -0.3040 & 0.6457 & -0.2817 & 2.2208 & 1.1970 & 3.0574 & 0.0547 & 2.1977 \\
0.0916 & -1.0241 & 0.9750 & 4.3330 & -0.7611 & -2.0215 & 1.6059 & 2.6500 & -0.3017 & -2.6130 \\
\\
2.2885 & 2.6082 & 0.8475 & 0.8422 & 0.3346 & -2.0668 & -0.1648 & -0.6743 & 0.7580 & -2.0629 \\
1.0597 & -0.0841 & 2.5858 & 0.2203 & 2.2596 & 0.6181 & -0.2308 & 0.2417 & 2.9146 & 2.1288
\end{array}\right] \tag{8.13}
$$

$$
y_{kr} = \begin{bmatrix}
-0.5044 & -1.3395 & -3.4283 & -1.2773 & 0.2762 & 0.2435 & -1.1202 & -2.8371 & -1.2826 & 1.0874 \\
1.1207 & -0.2399 & -3.9807 & -0.0911 & 0.0299 & 0.3958 & 1.0281 & 3.3078 & 1.5941 & 4.4146 \\
-0.4397 & -0.2541 & -1.1781 & 0.0110 & 2.4442 & 1.8911 & 3.4323 & -1.9287 & -0.5166 & 0.4103 \\
0.4315 & -2.5749 & 0.8496 & -1.0440 & 0.9247 & -2.2803 & 3.4038 & 0.8388 & -0.9080 & 2.3176 \\
-2.0918 & -0.1701 & -1.7809 & -0.1943 & 1.4228 & -2.5791 & -3.1986 & 0.7676 & 1.6528 & -0.0720 \\
0.4113 & 3.5843 & -0.1330 & -2.8352 & -1.3569 & -0.5588 & -0.6547 & -1.1289 & -1.0065 & 1.6756 \\
-0.7620 & 0.7231 & -1.1194 & 2.6888 & -0.5005 & 1.0038 & 3.6803 & 1.5162 & 1.0141 & 0.5096 \\
-2.1049 & 0.1269 & -3.5037 & -1.0527 & -1.8449 & -1.5016 & -0.0154 & 2.1691 & 0.9814 & 1.6436 \\
0.2747 & 2.0207 & 0.0250 & -1.2206 & 1.1275 & 0.9472 & 1.2434 & -1.1506 & 0.3399 & 0.2361 \\
-1.6078 & -1.6773 & 3.1017 & 2.6298 & 1.7364 & -1.6564 & 2.5967 & 3.3375 & -0.6210 & -2.8115 \\
0.4756 & 2.2140 & 1.5671 & 0.8103 & -0.4856 & 2.2005 & -0.5141 & 1.3590 & -0.3384 & 0.7940 \\
0.1598 & -0.3828 & -1.4584 & 0.8289 & 2.2686 & -3.8373 & 0.2511 & 2.2437 & 1.5274 & 0.7141 \\
1.0903 & 0.7856 & 1.4489 & -1.7976 & -2.4372 & -0.1319 & -3.2233 & 0.4164 & 0.4520 & -2.8987 \\
3.7455 & -6.4928 & -4.0527 & 1.4615 & -1.0884 & -0.8997 & -3.1751 & -0.5649 & -1.0778 & 1.3108 \\
1.8207 & -1.4158 & -0.4605 & 1.4743 & 0.7018 & 1.6292 & -1.3818 & -1.4263 & -1.5924 & 0.7265 \\
0.3706 & -1.3458 & -2.2328 & 1.1639 & -1.2430 & -1.7374 & -2.3665 & -1.3927 & -4.6119 & -1.8760 \\
1.6564 & -0.2804 & -1.6975 & -0.5116 & 0.2885 & 2.0118 & 0.9801 & 3.4138 & -1.1237 & 1.1336 \\
-0.7100 & -2.0331 & -0.5377 & 2.5779 & 0.2582 & -2.1331 & 0.0498 & 1.4617 & -0.5812 & -2.8397 \\
1.5261 & -0.3588 & -1.3557 & 1.3726 & 1.0956 & -0.4870 & 1.5799 & 0.9448 & -0.3009 & -0.0816 \\
-0.5569 & -1.5409 & 1.6963 & 2.5769 & 3.0259 & 1.5675 & 0.7857 & 0.6168 & 2.6494 & 3.6240
\end{bmatrix} \tag{8.14}
$$

Saída do N4SID para o sistema combinado:

$$y_{krc} = \begin{bmatrix}
1.5202 & -8.6449 & 7.5771 & -1.9236 & 2.4628 & 0.6472 & 2.6936 & -4.298 & -3.473 \\
-0.3273 & -10.6316 & -0.3486 & -3.6489 & 0.5742 & -1.7039 & 0.9696 & 3.0080 & 13.7232 \\
\\
-10.79 & -9.8640 & 1.6543 & -6.4185 & -1.1445 & 2.2082 & 2.6319 & 8.7154 & -3.3942 \\
14.0211 & 2.2932 & 6.4084 & 7.8194 & -0.7502 & -2.6117 & -7.7428 & -4.5418 & -8.9510 \\
\\
6.5563 & -15.6560 & 1.1957 & -3.3237 & -9.4211 & 2.6006 & -0.2866 & 0.8048 & 7.6612 \\
14.4089 & 5.8368 & 0.7389 & 12.8966 & 4.3684 & 1.4395 & -0.7167 & -11.0376 & 0.1251 \\
\\
-4.5981 & -0.3926 & 5.6960 & 1.5989 & 0.4727 & 2.7910 & 7.8070 & 0.2233 & 16.1497 \\
4.3111 & -6.2426 & -5.4618 & 1.3832 & -1.2854 & -6.9323 & -2.2433 & -17.1364 & 3.6261 \\
\\
-4.9462 & 6.5224 & 3.8593 & -7.2487 & 2.7609 & -5.6515 & 4.8114 & 0.8348 & -6.5662 \\
-5.0604 & -7.3294 & 3.9084 & -2.1207 & 8.0019 & -4.6404 & -2.5684 & 8.2382 & 10.3423 \\
\\
-5.9326 & -1.3622 & -9.2755 & -5.0781 & -13.6708 & 5.2066 & 2.2035 & -4.6999 & 7.7745 \\
6.5484 & 6.9432 & 7.2293 & 17.8147 & -1.4337 & -4.9243 & 6.3575 & -1.0703 & 2.1945 \\
\\
-5.5171 & 4.8053 & -5.0378 & 11.0337 & -19.2780 & 7.2701 & -6.5762 & 1.2615 & -0.2286 \\
-2.4109 & 3.4655 & -10.5993 & 17.3331 & -0.7447 & 2.5744 & 2.3953 & 2.6099 & -10.2484 \\
\\
7.3922 & 2.5654 & -5.2771 & -12.1971 & 12.2573 & -4.5082 & 9.9241 & 5.9585 & -6.5071 \\
-10.8046 & 6.8503 & 17.0744 & -6.1638 & -0.5479 & -11.5679 & -7.8321 & 4.8679 & -21.5902 \\
\\
20.6092 & -3.9313 & 10.3779 & 7.5590 & -4.9750 & 13.1970 & -1.9795 & 6.7542 & 3.7414 \\
-5.7072 & -15.0544 & -14.0915 & 1.0977 & -13.8816 & -3.5169 & -3.4089 & -6.6442 & -15.0815 \\
\\
11.6749 & 4.8571 & -1.4925 & -4.8366 & 5.2359 & -2.1979 & 11.5797 & -2.5650 & 4.9408 \\
-9.0631 & -2.4520 & 7.8951 & -1.4540 & -1.7452 & -11.9738 & -1.8558 & -3.9183 & -0.8337 \\
\\
1.4961 & 8.4277 & 1.7902 & 2.6671 & -1.6996 & -7.3447 & -0.0285 & -11.5114 & 1.1058 \\
-7.6020 & -7.0812 & -3.8825 & 2.9675 & 7.6036 & 3.8013 & 14.3538 & 4.8070 & 10.8235
\end{bmatrix} \tag{8.15}$$

Saída Benchmark com ruído branco na entrada correspondente ao Exemplo 1 do Capítulo 3,

específicamente na proposta de Algoritmo Combinado *AVW*.

$$y_k^d = \begin{bmatrix}
-0.5537 & -3.9195 & -0.0958 & -0.9764 & -0.2908 & 1.1389 & 0.9763 & -1.6114 & -1.4008 & -1.5972 \\
-0.7223 & -1.2476 & -0.2119 & 0.1566 & 2.0280 & 0.5540 & 0.3507 & 3.2174 & 0.8360 & 0.9857 \\
-2.0108 & -0.1941 & -2.1313 & 1.3639 & 1.5156 & 1.3588 & 4.0991 & -1.8865 & -0.4393 & -2.2868 \\
-0.0326 & -1.5771 & -1.9235 & 0.0392 & 0.2584 & -1.3898 & 2.1363 & 0.4121 & -1.7802 & 1.3922 \\
-0.7101 & -1.9535 & -0.9589 & 1.7356 & 0.9987 & 1.0559 & 0.7291 & -1.3540 & -0.1505 & 2.0294 \\
1.5166 & 4.2472 & -3.3106 & -1.9314 & -1.5951 & 0.4754 & -2.8930 & -2.0116 & -1.0753 & 1.5650 \\
0.4156 & 2.0827 & 1.9310 & 3.9630 & 1.0073 & 3.3866 & 2.2384 & 0.6420 & 2.0975 & -1.1037 \\
-0.6213 & 0.2271 & -2.6920 & -0.7703 & 0.1677 & 0.4865 & 1.2576 & 2.7664 & -0.7944 & 3.3270 \\
-0.0236 & 1.2124 & 0.6526 & 1.5423 & -1.2424 & -0.0362 & 1.0150 & -2.8388 & -2.4751 & -0.7473 \\
-2.3174 & -1.7541 & 3.8124 & 2.6741 & 0.8560 & -5.0057 & 0.9910 & 3.3613 & -2.1057 & -5.0581 \\
-0.4022 & 1.3705 & 0.8557 & 4.0825 & -0.4645 & 1.7550 & -0.6326 & 1.5033 & -2.1148 & 1.3937 \\
1.5212 & -0.0364 & 0.1093 & 0.9322 & 0.1699 & -4.2245 & 2.1017 & 2.9233 & 0.1159 & -0.9085 \\
-1.1819 & -0.2784 & 2.0743 & -1.0774 & -2.1192 & -0.1909 & -2.0183 & 3.2035 & -0.0286 & 0.4730 \\
1.8678 & -6.6686 & -3.0675 & 1.6294 & -0.6727 & -1.1887 & -2.1492 & -1.7305 & -0.6541 & 3.6261 \\
2.0728 & -1.5002 & 1.9832 & -0.2698 & -1.8659 & 1.3566 & -1.0816 & 0.5214 & -0.1024 & 1.1302 \\
-0.2576 & -1.8178 & -1.6217 & 0.8899 & -0.3995 & -0.8217 & 0.5898 & 0.5330 & -3.9239 & -2.5882 \\
1.3138 & 2.4137 & 0.6370 & -0.3464 & -0.0697 & 2.4587 & 0.1893 & 2.3154 & 1.1370 & 2.0662 \\
-0.3734 & -0.6532 & 1.7033 & 6.4452 & -2.1185 & -3.0442 & 2.6438 & 2.2602 & -1.6829 & -2.2975 \\
2.6784 & 2.6962 & 0.2119 & 0.2827 & 0.7782 & -3.0167 & 0.6164 & -0.1053 & -0.0637 & -2.3285 \\
2.6129 & 0.6238 & 4.5432 & 0.7247 & 4.1240 & 0.2784 & -1.3707 & 0.0307 & 4.1047 & 1.0126
\end{bmatrix} \tag{8.16}$$

Saída Benchmark com ruído colorido na entrada correspondente ao Exemplo 2 do Capítulo 3,

específicamente na proposta de Algoritmo Combinado *AVW*.

$$
y_{kc} = \left[\begin{array}{cccccccccc}
-0.521 & -4.669 & 0.2148 & -1.754 & -0.529 & 0.3913 & 0.4949 & -2.145 & -2.395 & -3.842 \\
-0.669 & -0.907 & 0.6233 & 0.1784 & 2.0516 & 0.0522 & -0.167 & 2.4851 & -0.275 & -0.848 \\
 & & & & & & & & & \\
-4.290 & -1.623 & -4.680 & -0.372 & 0.8615 & 0.9395 & 5.2032 & -1.485 & 0.4140 & -3.773 \\
-1.480 & -2.891 & -3.780 & -1.212 & -0.410 & -1.778 & 2.1583 & 0.5875 & -1.626 & 0.6179 \\
 & & & & & & & & & \\
-0.645 & -2.306 & -2.992 & 0.7887 & 0.1035 & 1.1403 & 1.3224 & -1.876 & 0.1279 & 3.3485 \\
1.6573 & 3.5154 & -4.467 & -2.639 & -2.430 & 0.2636 & -2.702 & -2.277 & -0.794 & 2.1542 \\
 & & & & & & & & & \\
1.2333 & 2.9736 & 2.9962 & 5.6668 & 2.5956 & 6.5050 & 3.2699 & 3.5807 & 4.0295 & 0.0895 \\
-0.2825 & 0.6539 & -1.8579 & 0.2261 & 1.6489 & 2.4341 & 2.2521 & 4.6977 & 0.2274 & 3.9997 \\
 & & & & & & & & & \\
1.2504 & 1.4935 & 2.7532 & 2.6837 & -0.721 & -0.586 & 1.4526 & -2.924 & -2.841 & -2.674 \\
-1.443 & -1.466 & 4.7871 & 2.9334 & 0.6268 & -5.333 & 0.5722 & 2.7391 & -2.958 & -5.987 \\
 & & & & & & & & & \\
0.5188 & 1.6027 & 0.6412 & 5.2913 & -0.4093 & 2.5785 & 0.1006 & 3.6478 & -3.2389 & 2.8157 \\
1.8537 & -0.3254 & -0.1415 & 1.0988 & 0.0273 & -3.7812 & 2.5793 & 3.4200 & -0.5370 & -0.1110 \\
 & & & & & & & & & \\
-1.2616 & -0.5550 & 1.8292 & 0.7660 & -1.4627 & -1.0572 & -3.9254 & 3.8387 & -0.9639 & 1.6117 \\
1.7109 & -6.5318 & -2.6921 & 2.7689 & -0.3563 & -1.8185 & -2.4410 & -1.2869 & -0.8878 & 4.5447 \\
 & & & & & & & & & \\
2.2265 & -2.0325 & 3.9538 & -0.3148 & -0.2035 & 2.1222 & -1.5331 & 1.8060 & -1.2881 & 1.1813 \\
-0.0247 & -1.0861 & 0.0907 & 1.6374 & 1.4755 & 0.0313 & 1.2180 & 1.7303 & -3.8084 & -2.0606 \\
 & & & & & & & & & \\
1.3726 & 3.5892 & 1.5216 & 0.3214 & -0.4510 & 2.8400 & 0.6976 & 4.7400 & 1.8031 & 3.4063 \\
0.2196 & 0.3589 & 2.1591 & 6.6273 & -2.1090 & -2.8046 & 3.1147 & 3.2520 & -1.3556 & -1.3625 \\
 & & & & & & & & & \\
4.2857 & 5.2007 & 2.6456 & 2.4109 & 2.2472 & -2.5568 & 0.9166 & -1.2493 & 0.3892 & -3.4131 \\
3.6473 & 2.2184 & 6.0415 & 2.1600 & 4.8693 & 0.7620 & -1.4473 & -0.7343 & 3.5152 & -0.6442
\end{array}\right] \tag{8.17}
$$

Controlador Gain-Scheduled Neural

Resultados da lei de controle para a entrada-referência (5,1)

$$u_k = \begin{bmatrix} 3.7307 & 3.0987 & 3.4031 & 3.2577 & 3.3273 & 3.2938 & 3.3101 & 3.2999 & 3.3084 & 3.3068 \\ 0.3779 & 0.0944 & 0.1735 & 0.1452 & 0.1566 & 0.1512 & 0.1534 & 0.1523 & 0.1524 & 0.1513 \end{bmatrix} \begin{bmatrix} 3.3063 & 3.3055 & 3.3058 & 3.3068 & 3.3081 & 3.3071 & 3.3082 & 3.3055 & 3.3066 & 3.3038 \\ 0.1515 & 0.1514 & 0.1513 & 0.1509 & 0.1505 & 0.1503 & 0.1501 & 0.1501 & 0.1501 & 0.1500 \end{bmatrix} \begin{bmatrix} 3.3070 & 3.3073 & 3.3050 & 3.3089 & 3.3107 & 3.3080 & 3.3114 & 3.3103 & 3.3106 & 3.3092 \\ 0.1498 & 0.1492 & 0.1493 & 0.1489 & 0.1481 & 0.1481 & 0.1478 & 0.1474 & 0.1473 & 0.1472 \end{bmatrix} \begin{bmatrix} 3.3111 & 3.3080 & 3.3113 & 3.3100 & 3.3107 & 3.3053 & 3.3152 & 3.3100 & 3.3114 & 3.3114 \\ 0.1470 & 0.1470 & 0.1468 & 0.1464 & 0.1463 & 0.1467 & 0.1459 & 0.1453 & 0.1457 & 0.1453 \end{bmatrix} \begin{bmatrix} 3.3110 & 3.3101 & 3.3124 & 3.3123 & 3.3096 & 3.3085 & 3.3104 & 3.3117 & 3.3101 & 3.3125 \\ 0.1452 & 0.1452 & 0.1449 & 0.1445 & 0.1447 & 0.1450 & 0.1447 & 0.1442 & 0.1441 & 0.1439 \end{bmatrix} \begin{bmatrix} 3.3138 & 3.3142 & 3.3131 & 3.3104 & 3.3153 & 3.3091 & 3.3168 & 3.3088 & 3.3156 & 3.3141 \\ 0.1433 & 0.1431 & 0.1430 & 0.1434 & 0.1429 & 0.1430 & 0.1427 & 0.1426 & 0.1426 & 0.1419 \end{bmatrix} \begin{bmatrix} 3.3152 & 3.3115 & 3.3158 & 3.3184 & 3.3128 & 3.3154 & 3.3141 & 3.3121 & 3.3174 & 3.3162 \\ 0.1419 & 0.1421 & 0.1419 & 0.1410 & 0.1413 & 0.1415 & 0.1413 & 0.1416 & 0.1411 & 0.1405 \end{bmatrix} \begin{bmatrix} 3.3151 & 3.3147 & 3.3150 & 3.3168 & 3.3150 & 3.3127 & 3.3157 & 3.3132 & 3.3124 & 3.3146 \\ 0.1408 & 0.1409 & 0.1408 & 0.1404 & 0.1404 & 0.1408 & 0.1406 & 0.1405 & 0.1408 & 0.1406 \end{bmatrix} \begin{bmatrix} 3.3162 & 3.3172 & 3.3157 & 3.3153 & 3.3142 & 3.3175 & 3.3147 & 3.3175 & 3.3143 & 3.3160 \\ 0.1401 & 0.1398 & 0.1398 & 0.1400 & 0.1401 & 0.1398 & 0.1397 & 0.1397 & 0.1397 & 0.1398 \end{bmatrix} \begin{bmatrix} 3.3136 & 3.3143 & 3.3161 & 3.3111 & 3.3180 & 3.3127 & 3.3175 & 3.3136 & 3.3175 & 3.3158 \\ 0.1399 & 0.1400 & 0.1397 & 0.1401 & 0.1398 & 0.1396 & 0.1397 & 0.1396 & 0.1395 & 0.1393 \end{bmatrix} \begin{bmatrix} 3.3168 & 3.3112 & 3.3169 & 3.3120 & 3.3167 & 3.3133 & 3.3148 & 3.3148 & 3.3133 & 3.3154 \\ 0.1394 & 0.1400 & 0.1399 & 0.1398 & 0.1398 & 0.1397 & 0.1399 & 0.1398 & 0.1400 & 0.1399 \end{bmatrix} \begin{bmatrix} 3.3159 & 3.3124 & 3.3124 & 3.3165 & 3.3123 & 3.3179 & 3.3147 & 3.3168 & 3.3135 & 3.3134 \\ 0.1397 & 0.1401 & 0.1405 & 0.1400 & 0.1401 & 0.1400 & 0.1398 & 0.1399 & 0.1401 & 0.1405 \end{bmatrix} \begin{bmatrix} 3.3165 & 3.3127 & 3.3119 & 3.3170 & 3.3169 & 3.3172 & 3.3159 & 3.3162 & 3.3142 & 3.3162 \\ 0.1402 & 0.1404 & 0.1410 & 0.1405 & 0.1400 & 0.1401 & 0.1403 & 0.1405 & 0.1407 & 0.1408 \end{bmatrix} \begin{bmatrix} 3.3146 & 3.3089 & 3.3164 & 3.3159 & 3.3148 & 3.3138 & 3.3126 & 3.3143 & 3.3124 & 3.3147 \\ 0.1408 & 0.1418 & 0.1416 & 0.1409 & 0.1412 & 0.1415 & 0.1419 & 0.1419 & 0.1420 & 0.1421 \end{bmatrix} \begin{bmatrix} 3.3139 & 3.3106 & 3.3143 & 3.3183 & 3.3125 & 3.3147 & 3.3105 & 3.3137 & 3.3127 & -0.3477 \\ 0.1420 & 0.1426 & 0.1426 & 0.1418 & 0.1422 & 0.1428 & 0.1431 & 0.1433 & 0.1432 & -0.1813 \end{bmatrix} \tag{8.18}$$

8.4.1 Sistema Variante no Tempo

As Figuras 8.1, 8.2 e 8.3 apresentam, respectivamente, os sinais de saída do sistema variante no tempo para $k = 4$, $k = 7$ e $k = 10$.

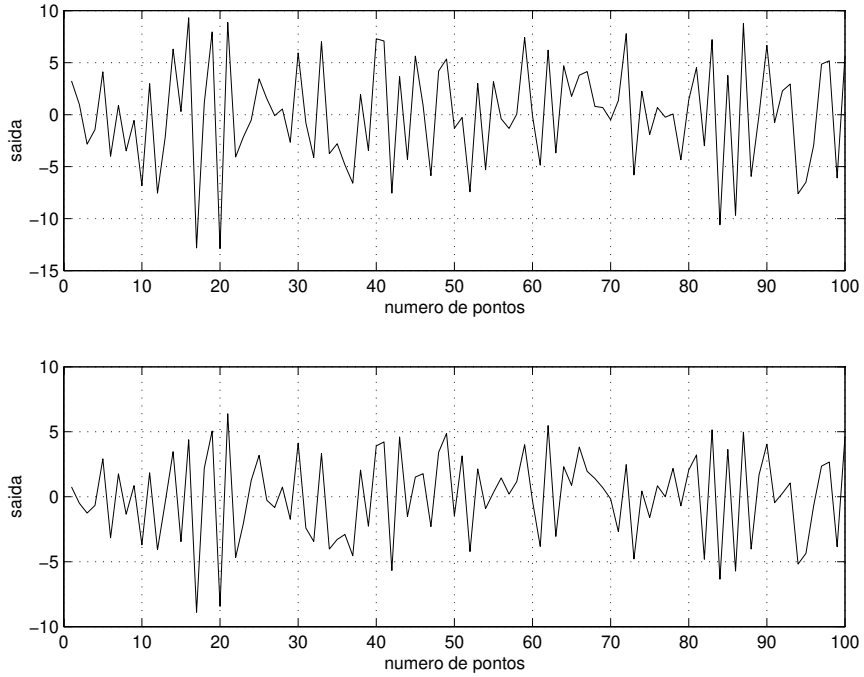

Fig. 8.1: Sinal de saída para k=4

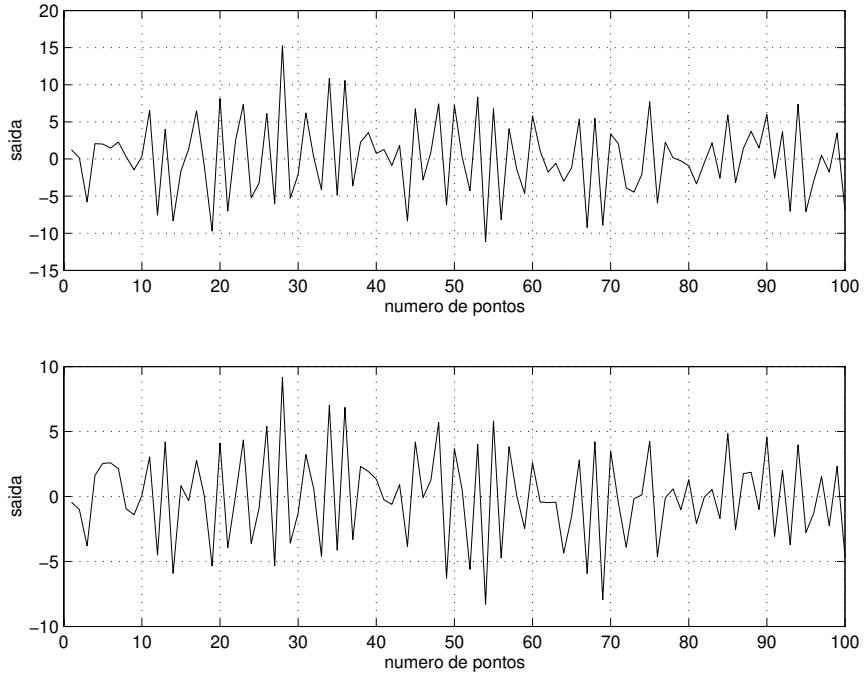

Fig. 8.2: Sinal de saída para k=7

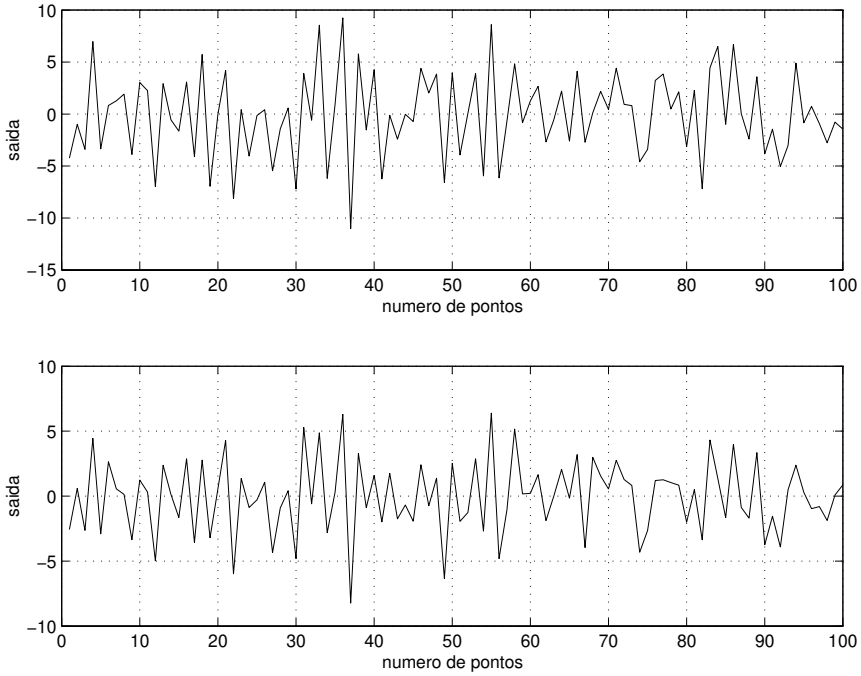

Fig. 8.3: Sinal de saída para k=10

Capítulo 9

Inteligência Computacional

A Inteligência Artificial (*IA*) é a parte da ciência da computação que agrega estruturas inteligentes aos sistemas computacionais tradicionais, isto é, acrescentam características que podem ser associadas com a inteligência do comportamento humano, como a linguagem natural (a falada), apreensão, raciocínio e resolução de um problema.

Solucionar problemas com computação convencional normalmente se restringe a solucionar problemas para os quais já foram determinados certos procedimentos, como os algoritmos matemáticos. No entanto alguns problemas do mundo real não permitem soluções determinísticas e seu processo de solução se constitui então numa estratégia de busca por soluções.

Um dos mais difíceis obstáculos a transpor quando se tenta aplicar técnicas de IA a problemas do mundo real está justamente relacionado com a magnitude e complexidade da maioria das aplicações - as inúmeras possibilidades de soluções envolvidas num problema. No início das pesquisas em IA, o objetivo principal era desenvolver bons métodos de busca para solucionar problemas até por conta das limitações dos computadores da época. De forma a estudar-se diferentes estratégias de busca por uma solução faz-se necessário antes buscar formas de representar o conhecimento existente sobre um certo domínio. E antes ainda necessitamos levantar conceitos fundamentais que passaremos a utilizar daqui por diante.

As metodologias empregadas para solucionar problemas na área da IA são mais heurísticas que sistemáticas ou algorítmicas. A seguir explicaremos em detalhe a diferença entre as duas metodologias.

Um *algoritmo* é uma regra matemática, ou uma lei, ou uma verdade que sempre que aplicada a premissas conhecidas, produz resultados senão conhecidos, ao menos esperados. Um algoritmo, programado num computador ou na mente humana, é uma solução lógica que pode ser verificável. Por exemplo, quando resolvemos uma equação de segundo grau, normalmente nos baseamos sempre nas mesmas fórmulas para encontrar suas raízes (sua solução), ou seja, estamos nos baseando num algoritmo.

A *heurística*, pelo contrário, é uma verdade cirscunstancial; que pode não ser verificável, não é matematicamente comprovável. Nesta técnica de resolver problemas, a solução é obtida através de tentativas e erros, ou, por seleção, conexão e mudanças associativas. Esta técnica é baseada naquilo que chamamos de regras práticas, baseadas em regras que o ser humano vai desenvolvendo no seu dia a dia, resultados da própria experiencia que vai adquirindo ao lidar com certas situações.

9.1 Objetivos da IA

Entre os objetivos que visam a utilização da *IA* temos o de permitir uma atuação eficiente sobre uma classe de problemas que não podem ser resolvidos pelos métodos computacionais convencionais, por exemplo:

- Problemas para os quais ainda não existem modelos matemáticos precisos, ou no caso de problemas para os quais não se pode determinar um modelo matemático;
- Problemas para os quais não se conhece uma solução algoritmica (determinística);
- Problemas relacionados com a compreensão da linguagem natural, exemplo um tradutor automático;
- Problemas relacionados com visão artificial, reconhecimento de padrões, caracterização de variáveis.

9.1.1 Ramos da IA

- Sistemas Especialistas, exemplo de apoio à tomada de decisão;
- Processamento de linguagem natural, exemplo reconhecimento de voz, uso de redes neurais artificiais;
- Robótica, exemplo no controle da trajetória de um robô, sincronismo dos robôs com outros sistemas;
- Aprendizado de máquina, exemplo redes neurais artificiais;
- Reconhecimento de padrões;
- Reconhecimento de impressões digitais, iris humana por exemplo.

9.2 Uma Introdução às Redes Neurais Artificiais (RNA)

Desde a metade da década de 50 pode-se distinguir as técnicas de IA em simbólicas e não simbólicas para simulação do raciocínio.

As técnicas tradicionais de processamento simbólico, utilizando regras e fatos, buscam através de dedução formal, simular o raciocínio humano, mas nem sempre é possível se alcançar a solução para determinado problema desta forma. Freqüentemente, experiências intuitivas (empíricas) que permitiram alcançar soluções de sucesso fazem parte da bagagem de conhecimentos humana.

A mais popular das técnicas não simbólicas é baseada em redes neurais ou sistema conexionista, constituindo por sua vez um novo paradigma metodológico no campo da *IA*, ou seja, no desenvolvimento de sistemas computacionais capazes de imitar tarefas intelectuais complexas, tais como a resolução de problemas, o reconhecimento e classificação de padrões, os processos indutivos e dedutivos, etc. As redes neurais imitam a estrutura física do cérebro como seu modelo base, isto é, na

maneira como o cérebro é organizado em sua arquitetura elementar, e em como a *RNA* é capaz de executar tarefas computacionais.

Da mesma maneira que no cérebro, as *RNA* são organizadas com um número de elementos individuais simples (os neurônios), que se interconectam unos aos outros, formando redes capazes de armazenar e transmitir informação provinda do exterior. Outra capacidade importante das *RNA* é a auto-organização ou plasticidade, ou seja, através de um processo de aprendizado, é possível alterar-se os padrões de interconexão entre seus elementos. Por este motivo, as *RNA* são um tipo de sistema conexionista, no qual as propriedades computacionais são resultado dos padrões de interconexão da rede.

Pesquisas em Neurobiologia têm comprovado que a plasticidade do sistema nervoso é uma característica única em relação a todos os outros sistemas orgânicos. Conforme deGroot, "a plasticidade neural é a propriedade do sistema nervoso que permite o desenvolvimento de alterações estruturais em resposta à experiência, como adaptação a condições mutantes e a estímulos repetidos".

Este fato é melhor compreendido através do conhecimento morfológico-estrutural do neurônio, da natureza das suas conexões sinápticas e da organização das áreas associativas cerebrais. Sem dúvida nenhuma "o aprendizagem pode levar a alterações estruturais no cérebro"(Kandel). A cada nova experiência do indivíduo, portanto, redes de neurônios são rearranjadas, outras tantas sinapses são reforçadas e múltiplas possibilidades de respostas ao ambiente tornam-se possíveis. Portanto, "o mapa cortical de um adulto está sujeito a constantes modificações com base no uso ou atividade de seus caminhos sensoriais periféricos".

As *RNA* foram desenvolvidas, originalmente, na década de 40, pelo neurofisiologista Warren McCulloch, do MIT, e pelo matemático Walter Pitts, da Universidade de Illinois, os quais, dentro do espírito cibernético, fizeram uma analogia entre células nervosas vivas e o processo eletrônico num trabalho publicado sobre "neurônios formais". O trabalho consistia num modelo de resistores variáveis e amplificadores representando conexões sinápticas de um neurônio biológico.

Desde então, mais enfaticamente a partir da década 80, diversos modelos de *RNA* têm surgido com o propósito de aperfeiçoar e aplicar esta tecnologia. Algumas destas propostas tendem a aperfeiçoar mecanismos internos da rede neural para aplicação na indústria e negócios, outras procuram aproximá-las ainda mais dos modelos biológicos originais.

As redes neurais trabalham de forma similar a maneira pela qual os neurônios do cérebro humano codificam informações, ao invés de serem programadas, elas são *ensinadas* para fornecer respostas aceitáveis.

As *RNA* consistem em um método de solucionar problemas de *IA*, construíndo um sistema que tenha circuitos que simulem o cérebro humano, inclusive seu comportamento, ou seja, aprendendo, errando e fazendo descobertas. São mais que isso, são técnicas computacionais que apresentam um modelo inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Uma grande *RNA* pode ter centenas ou milhares de unidades de processamento, enquanto que o cérebro de um mamífero pode ter muitos bilhões de neurônios. O processo de aprendizagem pode ser considerado uma importante indicação de inteligência, (senão a única, sob o ponto de vista de [103]), mas raramente tem sido obtido via programação de computadores. Conseguiu-se desenvolver algoritmos de aprendizado de sucesso através da simulação de modelos biológicos do cérebro - área conhecida como *Redes Neurais (RN)* ou *Conexionismo*.

O processo de comunicação entre os neurônios ocorre através de uma estrutura denominada *Sinapse*.

Apesar da complexidade da *RN* não permitir uma única definição, as linhas seguintes seguem como uma tentativa das inúmeras definições ou interpretações do que seja realmente uma *RN*:

Um grafo direcionado é um objeto geométrico que consiste de um conjunto de pontos, chamados nós, ao longo de um conjunto de segmentos de linhas direcionadas entre eles. Uma *RN* é uma estrutura de processamento de informação distribuída paralelamente na forma de um grafo direcionado, com algumas restrições e definições próprias, consistindo de nós com interconexões sinápticas e enlaces de ativação e esta caracterizada por quatro propriedades [57]:

1. Cada neurônio é representado por um conjunto de enlaces sinápticos lineares, um bias aplicado externamente e um enlace de ativação possivelmente não linear. O bias é representado por um enlace sináptico conectado a uma entrada fixa em $+1$.

2. As ligações sinápticas dos sinais de entrada com o respectivo peso do neurônio.

3. A soma ponderada dos sinais de entrada definem o campo local induzido do neurônio.

4. O enlace de ativação limita o campo local induzido do neurônio para produzir uma saída.

Os nós deste grafo são chamados elementos processadores. Suas arestas são conexões, que funcionam como caminhos de condução instantânea de sinais em uma única direção, de forma que seus elementos processadores podem receber qualquer número de conexões de entrada. Estas estruturas podem possuir memória local, e também possuir qualquer número de conexões de saída desde que os sinais nestas conexões sejam os mesmos. Portanto, estes elementos tem na verdade uma única conexão de saída, que pode dividir-se em cópias para formar múltiplas conexões, sendo que todos carregam o mesmo sinal.

Então, a única entrada permitida para a função de transferência (que cada elemento processador possui) são os valores armazenados na memória local do elemento processador e os valores atuais dos sinais de entrada nas conexões recebidas pelo elemento processador. Os únicos valores de saída permitidos a partir da função de transferência são valores armazenados na memória local do elemento processador, e o sinal de saída do mesmo. Sinais de entrada para uma *RN* chegam através de conexões que se originam do mundo externo, saídas da rede para o mundo externo são conexões que deixam a rede.

A funcão de transferência pode operar continuamente ou episodicamente. Sendo que no segundo caso, deve existir uma entrada chamada "ativam"que causa o ativamento da função de transferência com o sinal de entrada corrente e com valores da memória local, e produzem um sinal de saída atualizado (ocasionalmente alterando valores da memória). E no primeiro caso, os elementos estão sempre ativados, e a entrada "ativam"chega através de uma conexão de um elemento processador agendado que também é parte da rede.

De forma geral, a operação de uma célula da rede se resume em que os sinais são apresentados à entrada; cada sinal é multiplicado por um peso que indica sua influência na saída da unidade; e é feita a soma ponderada dos sinais que produz um nível de atividade; se este nível excede um limite (threshold) a unidade produz uma saída.

9.2.1 RNA - Inspiração Biológica

RNA são inspiradas biologicamente, um neurônio biológico consiste de um núcleo, axônio e dendritos.

As junções entre neurônios constituem as sinapses. Um neurônio artificial ou elemento processador (*EP*), modela os axônios e dendritos de sua contraparte biológica através das conexões ou sinapses, utilizando ponderações ou ajuste dos pesos.

O neurônio artificial é uma estrutura lógico-matemática que procura simular a forma, o comportamento e as funções de um neurônio biológico. Assim sendo, os dendritos foram substituídos por entradas, cujas ligações com o corpo celular artificial são realizadas através de elementos chamados de peso (simulando as sinapses). Os estímulos captados pelas entradas são processados pela função soma, e o limiar de disparo do neurônio biológico foi substituído pela função de transferência.

Todos os sinais que constantemente chegam a um neurônio via dendritos (são considerados como canais de entrada) são somados, e quando esta soma atinge um certo limiar, faz com que o neurônio dispare um sinal para os outros, via axônio (é comparado a um canal de saída). A soma dos sinais é realizada no núcleo. Através de sinapses (conexões com outros neurônios), os dendritos recebem sinais inibitórios ou excitatórios. As sinapses regulam a forma em como a informação passa pelos neurônios; explicado biologicamente isto é realizado através de uma forma de comunicação química realizada por transmissores químicos, denominados neurotransmissores.

Cada célula nervosa possui um "encaixe"de entrada para suas informações, ou seja, não é qualquer informação presente na sua entrada que se propaga para a saída. É necessário uma combinação correta de receptores nervosos com transmissores nervosos. Uma vez que este encaixe tenha sido feito de maneira eficaz, esta célula dispara, ou seja, propaga a informação para o próximo neurônio, liberando seus neurotransmissores. A eficácia deste encaixe é determinada pela presença em quantidade suficiente do neurotransmissor, a saída de um neurônio pode excitar ou inibir a entrada de outro, numa seqüência complexa de encaixes.

Numa rede neuronal biológica, as várias células nervosas que a compõem, ajustam suas sinapses primeiramente baseado no seu **genótipo** (informações presentes no DNA das células de qualquer organismo vivo desde o momento de seu nascimento) e posteriormente através do treino ou aprendizado ao longo da sua vida. No decorrer da vida de um organismo vivo, novas conexões são realizadas enquanto outras são ajustadas (de modo inibitório ou excitatório), caracterizando o que se conhece como o **aprendizado**.

Combinando diversos neurônios artificiais podemos formar o que é chamado de *RNA*. As entradas, simulando uma área de captação de estímulos, podem ser conectadas em muitos neurônios, resultando, assim, em uma série de saídas, onde cada neurônio representa uma saída. Essas conexões, em comparação com o sistema biológico, representam o contato dos dendritos com outros neurônios, formando assim as sinapses. A função da conexão em si é tornar o sinal de saída de um neurônio em um sinal de entrada de outro, ou ainda, orientar o sinal de saída para o mundo externo (mundo real). As diferentes possibilidades de conexões entre as camadas de neurônios podem gerar **n** números de estruturas diferentes.

As variantes de uma *RN* são muitas, e combinando-as, podemos mudar a arquitetura conforme a necessidade da aplicação, ou ainda, conforme o gosto do projetista. Basicamente, os itens que compõem uma *RN* e, portanto, sujeito a modificações, são os seguintes :

- conexões entre camadas

- camadas intermediárias
- quantidade de neurônios
- função de transferência
- algoritmo de aprendizado

9.2.2 O Elemento Processador (EP)

Um modelo de *RNA* é caracterizado pelos seus neurônios (ou *Elementos Processadores*.) isoladamente, as conexões entre eles (arquitetura ou topologia da rede) e seu esquema de aprendizado.

Um neurônio é uma unidade de processamento de informação que é fundamental para o desenvolvimento de uma *RN*. O modelo de um neurônio forma a base para o projeto de uma *RNA* e possue 3 elementos básicos:

Conjunto de enlaces sinápticos ou conexões: Um sinal x_j na entrada sináptica j conectado ao neurônio k é multiplicado pelo peso sináptico w_{kj} (primeiro subíndice refere-se ao neurônio e o segundo a entrada final sináptica para a qual o peso faz referência).

Um somatóriao para somar o sinal de entrada vezes o respectivo peso sináptico do neurônio.

Função ativação para limitar a amplitude da saída do neurônio.

Uma *RNA* pode ser comparada a um grafo orientado composto por um certo número de nós ou elementos processadores interconectados que operam em paralelo. Cada *EP* (ou neurônio artificial) possui um certo número de entradas e somente um único sinal de saída que se propaga através das conexões com os outros elementos processadores. A cada entrada de um *EP* está associado um peso sináptico que pode ser excitatório (positivo) ou inibitório (negativo), normalmente variando de -1 à $+1$. O sinal de entrada pode assumir uma variação contínua (desde -1 até $+1$) ou discreta (restrito aos valores binários ou 0 ou 1). Um valor contínuo poderia significar o grau de veracidade (ou possibilidade) associado a uma entrada, e no caso discreto, se a entrada é falsa ou verdadeira.

O *EP* avalia seus sinais de entrada realizando um somatório ponderado das suas entradas (através dos pesos sinápticos associados a cada entrada). Em termos matemáticos, podemos descrever o neurônio k pela equação:

$$\begin{aligned} u_k &= \sum_{j=0}^{N} w_{kj} x_j \\ x_0 &= +1, \\ w_{k0} &= b_k \end{aligned} \tag{9.1}$$

onde u_k representa a soma ponderada dos N sinais de entrada do neurônio k, w_{kj} representa o valor do peso sináptico associado a cada sinal de entrada do neurônio k e x_i representa os sinais de entrada, b_k é o bias. Os sinais são 1 ou -1. O neurônio calcula u_k e compara este a um valor thresholh (T). Se u_k é maior que T a saída é igual a 1, contrariamente seria igual a -1.

De maneira mais simplificada, isto significa somar todos os sinais de entrada que chegam a um neurônio levando em consideração o peso das conexões envolvido em cada sinal de entrada. O sinal

de saída do *EP* é encontrado aplicando-se ao somatório ponderado das suas entradas numa função ativação que determinará seu valor de saída (nível de ativação):

$$y_k = f(u_k) \tag{9.2}$$

onde f é a função de ativação do neurônio k.

Esta função pode caracterizar o neurônio em linear ou não linear. Um modelo não-linear simples é a função Degrau Lógico, neste caso quando o somatório ponderado de suas entradas atinge um certo valor (**threshold**), normalmente zero, este neurônio "dispara"ou é ativado - este modelo simples é também conhecido como *perceptron* ou neurônio binário.

A função *Sigmóide* é normalmente a mais utilizada nos neurônios não lineares, a sua saída é proporcional à soma ponderada das suas entradas, e é a que mais se aproxima da função ativação de um neurônio real. É definida como uma função estritamente crescente, exemplo:

Função logística:

$$f(u) = \frac{1}{1+\exp(-au)} \tag{9.3}$$

$$f^{'}(u) = \frac{a\exp(-au)}{[1+\exp(-au)]^2} \tag{9.4}$$

onde a é um parâmetro variável.

Função Tangente Hyperbólica, é normalmente usada em aplicações de modelagem e controle, definida como:

$$f(u) = \tanh(u) \tag{9.5}$$

$$\tanh(u) = \frac{1-\exp(-2u)}{1+\exp(-2u)} \tag{9.6}$$

a derivada da função ativação $f' = 1 - f^2$. Os neurônios da camada de entrada tem uma função ativação linear.

9.3 Principais Arquiteturas de RNA utilizadas para Modelagem e Controle

Uma *RN* completa é organizada na forma de camadas, pode possuir *n* neurônios na camada de entrada, *m* neurônios na camada seguinte e assim sucessivamente até a camada final, denominada camada de saída. Uma rede com mais de uma camada pode ser caracterizada como uma *rede multicamada*.

A forma pela qual os neurônios estão conectados uns aos outros (*topologia* ou *arquitetura da rede*) causa um enorme efeito na operação da rede neural. As duas arquiteturas de *RNA* que são mais usadas para propósito de modelagem e controle são:

1. Perceptron Multicamada.

2. Rede de Função básica radial (*RBF*).

Perceptron Multicamada ou RN Feedforward Multicamada: É uma rede estática que consiste de uma camada de entrada, uma camada de saída e uma ou mais camadas ocultas conectadas em *feed-forward*, ou seja a saída de um neurônio não depende nunca dos valores anteriores, os seus sinais se propagam num único sentido e as saídas dependem somente dos sinais que estão chegando dos outros neurônios - não há laços neste sistema. Os nós fonte da camada de entrada da rede fornecem os respectivos elementos do vetor de entrada, o qual constitui o sinal de entrada aplicado aos neurônios na segunda camada (primeira camada oculta). O sinal de saída da segunda camada é usada como entrada na terceira camada e assim até o final da rede. O conjunto de sinais de saída dos neurônios na camada final (saída) da rede constitui a resposta da rede ao vetor de entrada fornecido pelo nó fonte na camada de entrada [57].

Rede de Funções Base Radiais (*RBF*). A rede consiste de uma camada oculta, um vetor de entrada e uma camada saída. Uma das diferenças básicas com o Perceptron Multicamada está na utilização da função ativação pelos neurônios da camada oculta: em muitos casos toma-se uma função Gaussiana:

$$f(u) = \exp(-\frac{v^2}{2\partial^2}) \tag{9.7}$$

onde $v = \|x - \ell\|$, que é dado geralmente pela distância euclidiana, x é o vetor de entrada e ℓ e ∂ representam o centro e a largura da função radial, respectivamente.

Nas *Redes Neurais Recorrentes (RNR)* as saídas dos neurônios são realimentados à rede, resultando num sistema dinâmico. Um exemplo simples desta rede é a Rede Hopfield, que no tempo discreto pode ser representada como:

$$x_{k+1} = \tanh(Wx_k) \tag{9.8}$$

onde $x_k \in R^n$ é o vetor do estado, $W \in R^{nxn}$ é uma matriz de pesos sináptica.

9.4 Aprendizado em Redes Neurais

As redes neurais podem ainda ser classificadas pelo seu algoritmo (ou regra) de aprendizado em supervisionado, não supervisionado, auto-organizadas, etc.

No método de *aprendizado supervisionado*, se fornece à rede pares entrada-saída casados, isto é, para cada entrada é apresentado a saída esperada e a rede monitora a si própria corrigindo associações incorretas através de um processo de realimentação pela rede (correção dos pesos sinápticos). Já uma *rede não supervisionada* não possui acesso à saída desejada, esta rede deve aprender por mecanismos de estímulo-reação, comparável a forma como as pessoas inicialmente aprendem uma linguagem: somente pela audição repetida de certas palavras em momentos particulares, as pessoas aprendem a fazer associações entre idéias e palavras. Neste caso não existe ninguém para indicar se a associação feita está correta.

9.4.1 O processo de aprendizado

Denomina-se algoritmo de aprendizado a um conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para

determinados modelos de redes neurais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados.

A rede neural baseia-se nos dados para extrair um modelo geral. Portanto, a fase de aprendizado deve ser rigorosa e verdadeira, a fim de se evitar modelos espúrios. Todo o conhecimento de uma rede neural está armazenado nas sinapses, ou seja, nos pesos atribuídos às conexões entre os neurônios.

Uma rede neural aprende modificando suas respostas conforme a entrada se modifique. Os pesos são ajustados conforme o método de aprendizado utilizado. O valor dos pesos sinápticos varia somente durante a etapa de treinamento da rede neural, sendo ajustados de forma a acumular mais conhecimento que fica distribuído pelos seus pesos sinápticos, no seu emaranhado de conexões.

Treinar uma rede neural é uma questão de ajuste de pesos, que pode ser feito tanto manualmente quanto automaticamente, através de algoritmos computacionais.

A capacidade de aprendizado, processamento e informação inteligente estocada numa rede neural é determinada pela sua arquitetura (topologia) das conexões da rede e pelo algoritmo de treinamento utilizado. O comportamento de uma rede é determinado pelos pesos das suas conexões, que são estabelecidos durante o processo de treinamento. As regras de aprendizado descrevem como cada neurônio deve interpretar a informação vinda dos outros neurônios à ele conectados e assim, qual sinal distribuir pelo restante da rede.

9.5 Desenvolvimento de Aplicações

Esta seção procura ilustrar os passos necessários para o desenvolvimento de aplicações utilizando *RNA*.

1. Coleta de dados e separação em conjuntos

   Os dois primeiros passos do processo de desenvolvimento de aplicações com *RNA* são a coleta de dados relativos ao problema e sua separação em um conjunto de treinamento e um conjunto de testes. Esta tarefa requer uma análise cuidadosa sobre o problema para minimizar ambiguidades e erros nos dados. Além disso, os dados coletados devem ser significativos e cobrir amplamente o domínio do problema; não devem cobrir apenas as operações normais ou rotineiras, mas também as exceções e as condições nos limites do domínio do problema.

   Normalmente, os dados coletados são separados em duas categorias: dados de treinamento, que serão utilizados para o treinamento da rede e dados de teste, que serão utilizados para verificar seu desempenho sob condições reais de utilização. Além dessa divisão, pode-se usar também uma subdivisão do conjunto de treinamento, criando um conjunto de validação, utilizado para verificar a eficiência da rede quanto a sua capacidade de generalização durante o treinamento, e podendo ser empregado como critério de parada do treinamento.

   Depois de determinados estes conjuntos, eles são geralmente colocados em ordem aleatória para prevenção de tendências associadas à ordem de apresentação dos dados. Além disso, pode ser necessário pré-processar estes dados, através de normalizações, escalonamentos e conversões de formato para torná-los mais apropriados à sua utilização na rede.

2. Configuração da rede

   O terceiro passo é a definição da configuração da rede, que pode ser dividido em três etapas:

- Seleção do paradigma neural apropriado à aplicação.
- Determinação da topologia da rede a ser utilizada - o número de camadas, o número de unidades em cada camada, etc.
- Determinação de parâmetros do algoritmo de treinamento e funções de ativação. Este passo tem um grande impacto no desempenho do sistema resultante.

Existem metodologias, "dicas" e "truques" na condução destas tarefas. Normalmente estas escolhas são feitas de forma empírica. A definição da configuração de redes neurais é ainda considerada uma arte, que requer grande experiência dos projetistas.

3. Treinamento

   O quarto passo é o treinamento da rede. Nesta fase, seguindo o algoritmo de treinamento escolhido, serão ajustados os pesos das conexões. É importante considerar, nesta fase, alguns aspectos tais como a inicialização da rede, o modo de treinamento e o tempo de treinamento.

   Uma boa escolha dos valores iniciais dos pesos da rede pode diminuir o tempo necessário para o treinamento. Normalmente, os valores iniciais dos pesos da rede são números aleatórios uniformemente distribuídos, em um intervalo definido. A escolha errada destes pesos pode levar a uma saturação prematura. Nguyen e Widrow encontraram uma função que pode ser utilizada para determinar valores iniciais melhores que valores puramente aleatórios.

   Quanto ao modo de treinamento, na prática é mais utilizado o modo padrão devido ao menor armazenamento de dados, além de ser menos suscetível ao problema de mínimos locais, devido à pesquisa de natureza estocástica que realiza. Por outro lado, no modo batch se tem uma melhor estimativa do vetor gradiente, o que torna o treinamento mais estável. A eficiência relativa dos dois modos de treinamento depende do problema que está sendo tratado.

   Quanto ao tempo de treinamento, vários fatores podem influenciar a sua duração, porém sempre será necessário utilizar algum critério de parada. O critério de parada do algoritmo backpropagation não é bem definido, e geralmente é utilizado um número máximo de ciclos. Mas, devem ser considerados a taxa de erro médio por ciclo, e a capacidade de generalização da rede. Pode ocorrer que em um determinado instante do treinamento a generalização comece a degenerar, causando o problema de over-training, ou seja a rede se especializa no conjunto de dados do treinamento e perde a capacidade de generalização.

   O treinamento deve ser interrompido quando a rede apresentar uma boa capacidade de generalização e quando a taxa de erro for suficientemente pequena, ou seja menor que um erro admissível. Assim, deve-se encontrar um ponto ótimo de parada com erro mínimo e capacidade de generalização máxima.

4. Teste

   O quinto passo é o teste da rede. Durante esta fase o conjunto de teste é utilizado para determinar o desempenho da rede com dados que não foram previamente utilizados. O desempenho da rede, medida nesta fase, é uma boa indicação de seu desempenho real.

   Devem ser considerados ainda outros testes como análise do comportamento da rede utilizando entradas especiais e análise dos pesos atuais da rede, pois se existirem valores muito pequenos,

as conexões associadas podem ser consideradas insignificantes e assim serem eliminadas (prunning). De modo inverso, valores substantivamente maiores que os outros poderiam indicar que houve over-training da rede.

5. Integração

   Finalmente, com a rede treinada e avaliada, ela pode ser integrada em um sistema do ambiente operacional da aplicação. Para maior eficiência da solução, este sistema deverá conter facilidades de utilização como interface conveniente e facilidades de aquisição de dados através de planilhas eletrônicas, interfaces com unidades de processamento de sinais, ou arquivos padronizados. Uma boa documentação do sistema e o treinamento de usuários são necessários para o sucesso do mesmo.

   Além disso, o sistema deve periodicamente monitorar sua performance e fazer a manutenção da rede quando for necessário ou indicar aos projetistas a necessidade de retreinamento. Outras melhorias poderão ainda ser sugeridas quando os usuários forem se tornando mais familiares com o sistema, estas sugestões poderão ser muito úteis em novas versões ou em novos produtos.

6. Exemplo de Implementação

   Para exemplificar o desenvolvimento de uma rede neural, tomemos o cálculo da função y como a raiz quadrada de x ($y = sqrt(x)$). Temos, então dois neurônios para a camada de entrada de dados, um de "bias" e outro de entrada efetiva; um neurônio para a saída e, três neurônios na camada oculta.

   Sejam os dados, números entre 1 a 100 com suas respectivas raízes quadradas. Serão escolhidos aleatóriamente 10 números para a fase de testes e os restantes para a fase de treinamento.

   O aprendizado começa com a aplicação de 5000 iterações à rede neural e em seguida é realizado o teste, onde se compara os resultados obtidos com os valores reais. A diferença encontrada nesta comparação define o grau de ajuste que os dados obtidos pela rede neural deve sofrer, em relação aos dados reais.

   Outras 5000 iterações são realizadas, dando segmento a fase de aprendizado, seguido de novos testes. Se a diferença entre os dados obtidos e os reais diminuiu, significa que o nível de aprendizado melhorou e que novas 5000 iterações serão aplicadas a fim de se refinar a rede. Caso contrário, a rede foi treinada em excesso, fazendo com que ela memorize os dados e não produza uma relação entre eles.

   Comparando com uma criança na escola, digamos que a rede neural "decorou" a lição, e não realmente "entendeu", "assimilou", cometendo erros em "exercícios" semelhantes aos que lhe foram apresentados, mas de valores alterados.

   Após 30000 iterações de aprendizado, a rede neural informou o valor 5,942 para a raiz quadrada de 36, ou seja, um erro de aproximadamente 1. Obviamente, para este caso, a rede neural não se mostrou mais eficiente que uma função $sqrt$ de qualquer linguagem estruturada, mas pode-se perceber o poder de aprendizado e de exatidão de uma rede neural, se devidamente treinada.

9.6 Redes Neurais para Identificação e Controle

Identificação e controle de sistemas dinâmicos são áreas da engenharia vastamente exploradas e com grande potencial de aplicação. As técnicas convencionais (métodos clássicos, por exemplo) são baseadas principalmente na teoria de sistemas lineares. A álgebra linear e as equações diferenciais lineares ordinárias são as ferramentas que caracterizam esse tipo de abordagem.

Porém, a aplicação dessas técnicas muitas vezes é limitada devido às condições de não-linearidade do sistema em questão ou do ambiente no qual ele está imerso. Para esses casos abordagens lineares podem não satisfazer de maneira completa os requisitos do sistema.

Uma solução possível seria o projeto ou análise de sistemas não-lineares de identificação e controle. Entretanto os métodos de projeto de sistemas não-lineares são muito específicos e, em alguns casos, inerentes ao processo em questão.

Diante da dificuldade da modelagem e controle de tais sistemas as *RNA*, com sua capacidade de modelar processos não-lineares complexos, passaram a ser abordadas como possível solução desse tipo de problema.

Esta pesquisa busca elucidar diversos pontos associados a controle e modelagem de sistemas utilizando redes neurais por meio da apresentação de referências a textos científicos relacionados com esse assunto.

Para facilitar a consulta e caracterizar melhor os temas abordados dentro dessa área, os textos foram agrupados em sub-items. Eles são:

- Modelos de redes neurais utilizadas em identificação e controle: Um dos principais pontos que deve ser abordado no projeto da rede neural é o modelo de rede a ser utilizado. Diversos modelos de *RNA* podem ser aplicados em problemas de identificação e controle de sistemas dinâmicos. Os textos citados neste sub-item apresentam alguns exemplos de utilização de modelos de redes tradicionais, como as MLP (multilayer perceptron), redes RBF (radial basis function) e redes de Hopfield. Além destes modelos tradicionais, alguns modelos propostos recentemente, como as redes CPBUM (Chebyshev Polynomials Based Unified Model), também foram utilizados no problema de identificação e controle.

- Comparações com técnicas convencionais de Identificação e controle: Uma das maneiras de se ter uma maior clareza sobre a eficiência da utilização de redes neurais é compará-las com outros tipos de ferramentas que apresentam finalidades comuns. Por exemplo, é possível fazer comparações de redes neurais (RBF e TDNN) com o controlador do tipo *PID*. Em [92] é feito um apanhado sobre a convergência de identificação de sistemas sobre redes neurais.

- Algoritmos de treinamento: Após a definição do modelo de rede utilizado para identificação e controle de sistemas, é necessário propor um algoritmo de treinamento para a rede neural. Os textos citados neste sub-item sugerem o paradigma de treinamento supervisionado, em que dados reais da planta a ser modelada ou controlada são fornecidos à rede de modo que o mapeamento não-linear possa ser realizado. Alguns textos sugerem uma variação do algoritmo tradicional de backpropagation, chamado de backpropagation dinâmico. Uma técnica de aprendizado denominada aprendizado ativo também é abordada. Neste tipo de aprendizado, a rede tem a capacidade de selecionar seus próprios dados de treinamento.

- Aplicações: Assim como todos os métodos de controle e identificação clássicos, o objetivo final para as redes neurais nessas áreas são as aplicações. Foram encontrados diversos artigos tratando de aplicações possíveis para controle e identificação. É possível observar pela leitura dos textos que o principal enfoque para as aplicações está relacionada com sistemas não-lineares. Isso ocorre devido à dificuldade de manipulação com esses sistemas por meio de ferramentas clássicas e, decorrente disso, ao forte atrativo oferecido pelas redes neurais para esse tipo de sistema.

- Configurações de controle e identificação: Uma das condições para que seja possível efetuar de maneira consistente o controle ou a identificação de um sistema é a correta inserção do sistema de controle ou de identificação ao sistema. A partir do momento que um sistema se insere e outro é gerado, na verdade um terceiro sistema. No caso de identificação passiva é interessante que o elemento identificador não interfira no sistema, se posicionando de maneira a não alterar o estado do sistema. Entretanto os métodos de identificação on-line e os sistemas de controle têm como principal intenção à interferência no sistema. Conforme o esperado, para cada tipo de atribuição a que se presta o sistema deve ser adota uma configuração específica. Todos os artigos referenciados tratam desse assunto indicando configurações possíveis para diversos tipos de sistemas.

- Técnicas de projeto de controladores neurais: Os textos citados neste sub-item têm por objetivo introduzir metodologias de análise e projeto de sistemas de controle não-lineares utilizando redes neurais artificiais, utilizando conceitos sólidos de engenharia de controle e estabelecendo conexões entre redes neurais e teoria de controle.

As *RNA* possuem muitas propriedades desejadas que os tornam adequadas para controle inteligente, a seguir serão numeradas algumas delas:

1. Elas aprendem por experiência e não por modelagem ou programação.
2. Elas possuem a habilidade de generalizar, isto é, mapear entradas similares para saídas similares.
3. Elas podem formar arbitrariamente mapeamentos não-lineares contínuos.
4. Elas tem arquitetura que são distribuídas, inerentemente paralelas e potencialmente em tempo real.

   Para *controle neuro fuzzy* propriedades adicionais são necessárias e serão apresentadas a seguir:

5. Estabilidade temporal, a habilidade de absorver nova informação (plasticidade), reter conhecimento (ou regras) previamente codificada através da rede ou na base de regras (estabilidade).
6. Adaptação em tempo real para variações paramétricas, isto é, aprendizado.
7. Provada as condições de convergência do aprendizado para otimização global e local.